論文 / 著書情報
Article / Book Information

題目(和文)	DLB下の作業共有環境における作業負荷の効果
Title(English)	The Effect of Workload in Work-sharing Environment under DLB Policy
著者(和文)	ｶｽﾓｻﾗｰ
Author(English)	Salah Kasmo
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第9891号, 授与年月日:2015年3月26日, 学位の種別:課程博士, 審査員:圓川　隆夫,伊藤　謙治,梅室　博行,青木　洋貴,鈴木　定省
Citation(English)	Degree:, Conferring organization: Tokyo Institute of Technology, Report number:甲第9891号, Conferred date:2015/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

The Effect of Workload in Work-sharing Environment under DLB Policy

Salah Kasmo

12D42035

Submitted in partial fulfillment of the requirement of the Degree of Doctor of Philosophy

Supervisor:

Prof. Takao Enkawa

Assoc. Prof. Sadami Suzuki

Department of Industrial Engineering and Management

Graduate School of Decision Science and Technology

Tokyo Institute of Technology

March 2015

TABLE OF CONTENTS

ABSTRACT

Work sharing has received a lot of interest whether in literature or application. The benefits that are derived from its applications since it came up in Japanese production system (Toyota Production System) encourage the researchers and practitioners to invent ways to apply it as suitable. One of the famous methods is Dynamic Line Balancing (DLB). DLB has an ability to manage the work sharing in the conventional serial line where workers are assigned to their own stations and no movement is allowed. The job has two types of task to be completed, fixed tasks done by unique workers and shared tasks done by the adjacent workers. To manage the work sharing in DLB, information about the system status is needed and it is usually done by a control rule. The worker after finishing his fixed task needs to make a decision to keep processing the shared task or send it to downstream buffer. In this sense, the size of fixed tasks or fixed workload can affect severely the performance as the shared task is managed by DLB. The most of interest was on the shared part of DLB research than other parts. Therefore, we explored the fixed workload, as it cannot be tackled after the process design is set.

The Toyota Production System is still the father of the most new work sharing initiatives. Our research has started by studying a divisional cell as a resembling example to DLB model. We introduced a helping zone concept to this cell. It is correspondent to the shared task in DLB. We investigated the effect of buffer on the cell performance where pioneer companies in cell manufacturing have begun inserting it after it was an absolute evil. The cases where it is useful or not have been specified.

To uncover the effect of workload, the experiments were done on a two-station line with a variety of workload configurations. To do the analysis, a workload measure has been

made up. The outcomes showed a special pattern of workload performance. This pattern changes as the number of jobs in the line changes. Moving to the factors affecting the performance in the environment, the pattern has a diversity of changes in term of value and trend.

Several factors have been subjected to experimenting and analysis. The factors are Information Accuracy, Granularity of shared task and Variability. Two levels of information accuracy have been tackled. The low level requires information about the downstream available work in the buffer while the high one needs besides to that the undone work at downstream stations. For granularity, the order of resulting subtasks and workload were tested. We also unveiled the variability's influence of both fixed and shared tasks unlike the previous papers which focused only on the shared work. The results highlight the areas where the improvement can give more revenue if it conducts firstly on these areas.

After the analysis of workload effect, we sought to consider the workload in the control rules that govern the work sharing in DLB. The rules where the available workload is used to find out the cutoff value are our target. The workload can be involved easily in these kinds of rules. Among these rules we chose a rule with the highest performance based on previous researches, which is HFB (Half Full Buffer). The ratio of fixed workload is inserted and a new rule is derived. We called it WFB (Workload-full-Buffer). In WFB, the available workload is divided by the fixed workload ratio instead of 2 as in HFB. In general, WFB showed a remarkably better performance among the rules used in this research which are the most near-optimal rules.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere thanks to Prof. Enkawa Takao for his guidance and support to do this. At the same time to Associate Prof. Suzuki Sadami who played an important role in this work and whose comments and help was valuable.

Big thank goes to my colleagues in Enkawa-Suzuki laboratory, who are wonderful friends and gave the support and encouragement when I needed.

Also I would like to thanks the Japanese Government represented by the Ministry of Education, which made possible my studies and economic support during my stay in Japan to complete the doctoral course.

Finally, I would like to express my greatest gratitude to my family in Syria and Japan, from whom I received encouragement and support, and their prayers gave me the power to overcome all the difficulties I faced.

List of Tables

Page

List of Figures

Page

CHAPTER 1

INTRODUCTION

Overview

This chapter presents the previous researches around the points tackled by this research and states the objectives this dissertation will treat with in the subsequent chapters. It also briefly clarifies the structure of this work.

1.1 Background

Work sharing has a lot of interest in recent decades whether in practical or academic environments. The advantages of work sharing include both operational and human-related. In the operational viewpoint, it helps to reduce the buffer size between stations, increases throughput and enhances the efficient utilization of capacity. For the human-related viewpoint, it alleviates the effects of repetitive work, combats boredom and raises the worker moral towards the work.

Applying work sharing needs cross-trained workers to do the shared work. There are three levels of cross training; no cross training, partially cross training and fully cross training. In the case of no cross training, each worker is assigned to a specific task or can attend a unique station and the work sharing is not allowed. This type can be found in the traditional serial lines. The other extreme case is full cross training where the worker can do any task of the job in collaborative or non-collaborative way [1]. This level gives a dramatic improvement in the

performance but the layout of line and the implementation mechanism must be considered well to achieve that improvement. On the other side, it is difficult to achieve and time- and cost-consuming. Such kind of cross training is applied in Japanese cell manufacturing (which is different from western cellular manufacturing in many aspects according to [2]) in a type of cells called yatai cell or single-worker cell where one worker completes the whole job [3]. Between these two extremes, partially cross training is the more reasonable and applicable solution. The worker here can do two types of tasks, fixed (unshared) and shared tasks.

Many academic researchers in recent years tackled the case with partially cross training. The best example is the divisional cell where each worker runs a set of stations or machines but here there is no work sharing between the adjacent workers [4], [5]. Cherry picking, two-skill chain and partial skill chain are three strategies considering work sharing to support the bottleneck station [6]. In cherry picking, all workers can helps the high-utilization stations directly. Meanwhile, in two-skill chain and partial chain, each worker can do his task and the task of his downstream station but in partial chain some workers can do only his own task. Two-skill complete chain is robust and efficient for implementing workforce agility in serial production lines [6], [7], [8]. Parvin et al. [9] introduced a new model of worker cross training, called a Fixed Task Zone chain (FTZC). This model is intended to employ in U-shaped Constant Work In Process lines. Each worker in this system is responsible for a set of stations and can share the work in two overlapping stations; one upstream of his zone and another downstream. They ended that FTZC can nearly reach the performance of a fully cross-trained system when it is designed using the ZonA algorithm. Among other ways of applying work sharing with partially cross training, Dynamic Line Balancing (DLB) policy [10] is one of the famous strategies. In DLB, the workers will stay at their stations and there is no movement between

stations. Each worker is assigned to fixed tasks of a job that can be done only by him or her and can help the upstream and downstream workers in shared tasks after finishing the fixed ones.

In work-sharing with limited cross-training, the decision when to do the shared task needs to be specified. In other words, when the worker has to pass on a job with the shared task undone or complete the shared task. There are two policies; the optimal policy and near optimal heuristic policy. The optimal policy has a complicated structure and it is difficult to apply in the [6], [11]. The near optimal heuristic policy mainly bases on threshold rules and it is easy to obtain and apply [10], [12], [13], (we adopted this type of policies). Ostolaza et al. [10] found that by using DLB with a half-full buffer (HFB) control rule, the Work-In-Process (WIP) inventory can be reduced and the efficiency can be improved. McClain et al. [12] demonstrated that DLB could increase the efficiency even with no buffer. They used a new model called subtask model where the tasks of a job are divided into *k* equal subtasks and they employed Erlang-*k* distribution to represents the task times.

The work sharing attacks the two types of workload imbalance. The first one is the imbalance resulting from the differences in the size of workload allocated for each work center. The second one is the imbalance resulting from the variability. So even if a line is balanced in related to the average load, it remains significantly unbalanced in the short term due to the variability [11]. The work sharing provides capacity buffering and variability buffering. As a result, it cuts the size of inventory required to keep the line flow smoothly and reach the targeted throughput. Reducing the inventory is considered the important goal for the companies as it has a group of serious negative effects in term of wasting the resources and hiding the chronic or temporary problems and others. In the Japanese mindset of manufacturing and quality practitioners, it is an absolute evil [14]. This idea has changed with some tolerance regarded

with accepting a small amount of inventory. This change comes because of the flood of diverse products with short cycle life. Therefore, the training for a wide range of skills is not feasible. Even though in short term, the high degree of work imbalance requires a high level of work [15], [16]. In the light of this situation, the compromise is done with a narrower scope of skills and as less as necessary of the inventory.

1.2 Purpose of This Dissertation

The focus of this research is on dynamic line balancing DLB as a mechanism of application the work sharing. The research papers in DLB have dealt with the workload imbalance as follows. Some concentrated on the variability-buffering role of work sharing and assume the workload is balanced and with some examples of an imbalanced line to show the capability of DLB in term of improving the performance [13]. Others counted the workload imbalance as a factor to generalize their resulting insights and treated in brief [11].

The workload is a dominant element on the performance even under application the work sharing. Of course, the moderate level of work sharing or partially cross training is applied here since it is more viable. Moreover, allocating the workload in equal portions between the work centers is not an easy mission in many cases due to technical or quality related issues. In the light of what mentioned above, this work investigates the workload effect in a serial production line with applying DLB policy as a work sharing mechanism. This also includes the interactions with other structural factors affecting the performance. Moreover, this research seeks to improve the performance resulting from DLB policy based on the above investigation.

1.3 Structure of This Dissertation

This dissertation has six chapters as follows.

Chapter 1 explores the previous researches done in the scope of this research. It states and clarifies the purpose of this work that tackles the important idea missed in the previous works.

Chapter 2 introduces DLB from Japanese Cell Manufacturing's viewpoint by inserting a buffer in a divisional cell. Then, it elucidates the role of buffer between stations on the performance. The analysis done under several factors ends to the cases where the buffer is needed to get a better performance.

Chapter 3 describes in more details the used work sharing policy (DLB). It illustrates the fundamental model that the whole research relies on with some deviations according to the investigated point. It shows the effect of workload imbalance on the performance and its trends.

Chapter 4 investigates several factors that influence the performance trend or even value of the assumed workload configurations. These factors are; Information Accuracy, Granularity of shared task and Variability.

Chapter 5 introduces a suggestion to improve the performance. It considers the workload imbalance studied in the previous chapters in modifying a famous control rule HFB.

Chapter 6 sums up the findings and contribution of this research. In addition, it highlights the important implications that can be extracted from the outcomes.

References

[1] Van Oyen, M.P., Gel, E.S. and Hopp, W.J., 2000. Performance opportunity of workforce agility in collaborative and noncollaborative work systems. *IIE Transactions,* 33(9): 761-777.

[2] Sakazume, Y. 2005. Is Japanese Cell Manufacturing a New System? A Comparative Study between Japanese Cell Manufacturing and Cellular Manufacturing, Journal of Japan Industrial Management Association, 55(6): 341–349.

[3] Yin, Y., Kaku, I., Murase, Y.,and Yasuda,Y., 2008, Seru (Cell) and Reverse Conversion: Part I. Definition, Self-evolution, and Typology, *Research Group of Economics and Management*, No. 2008-E01.

[4] Nakade, K. and Nishiwaki, R., 2008. Optimal allocation of heterogeneous workers in a U-shaped production line. *Computers and Industrial Engineering*, 54(3): 432-440.

[5] Nakade, K. and Ohno, K., 2003. Separate and carousel type allocations of workers in a U-shaped production line. *European Journal of Operational Research*, 145(2): 403-424.

[6] Hopp, W.J., Tekin, E. and Van Oyen, M.P., 2004. Benefits of skill chaining in serial production lines with cross-trained workers. *Management Science,* 50(1): 83-98.

[7] Jordan, W.C., Inman, R. R. and Blumenfeld, D.E., 2004. Chained cross-training of workers for robust performance. *IIE Transaction,* 36:953-967.

[8] Inman, R. R., Jordan, W.C. and Blumenfeld, D.E., 2004. Chained cross-training of assembly line workers. *International Journal of Production Research,* 42 (10): 1899-1910.

[9] Parvin, H., Van Oyen, M. P. and Pandelis, D. G., Williams, D. P. and Lee, J., 2012. Fixed task zone chaining: worker coordination and zone design for

inexpensive cross-training in serial CONWIP lines. *IIE Transactions,* 44: 894-914.

[10] Ostolaza, J., McClain, J. and Thomas, J., 1990. The use of dynamic (state-dependent) assembly-line balancing to improve throughput. *Journal of Manufacturing and Operation Managemen*t, 3:105-133.

[11] Gel, E. S., Hopp, W. J. and Van Oyen, M. P., 2002. Factors affecting opportunity of worksharing as a dynamic line balancing Mechanism. *IIE Transactions,* 34(10): 847-863.

[12] McClain, J.O., Thomas, J. and Sox, C., 1992. On-the-fly line balancing with very little WIP. *International Journal of Production Economics,* 27: 283-289.

[13] Chen, J., and Askin, R.G., 2006. Throughput maximization in serial production lines with worksharing. *International Journal of Production Economics,* 99: 88-101.

[14] Shingo, S., 1985. Non-stock production: the Shingo system for continuous improvement. *Productive Press*, pp.xx.

[15] Darwin, J. D., Hemant, V. K., and Bret J. W., 2009, "Influence of Workload Imbalance on the Need for Worker Flexibility," *Computer and Industrial Engineering*, 57: 319-329.

[16] Cesani, V., I. and Steudel, H. J., 2005. A study of labor assignment flexibility in cellular manufacturing systems. *Computers and Industrial Engineering*, 48: 571-591.

CHAPTER 2

HELPING ZONE AND BUFFER STATIONS

Overview

We introduce DLB the policy used in this research from the Japanese cell manufacturing's gate. DLB is almost a divisional cell with helping zones. Unlike cell manufacturing, DLB allows to have a buffer between stations. However, this idea is changing and cell manufacturing has started to be more tolerant with the buffer. The influence of buffer between stations is treated in this chapter. We clarify the complementary role of buffer on the performance under DLB (i.e. an divisional cell with helping zone and a buffer). Considering some structural factors of the line, the results display the favorite cases where the buffer boosts the performance and where it has no effect to add.

2.1 Introduction

The inventory between stations has generally a bad reputation among the practitioners and researchers of production and operations management. It has many disadvantages that touches several aspects of the production processes. Excessive inventory might cover the quality problems since there is not any warning sign to draw the attention like stopping the line. It also consumes a considerable area. From the financial view, it comprises a wasted asset and what is so-called the opportunity cost. It also prolongs the lead-time, which results a bad performance [1].

Therefore, the quest has been intensive to reduce it or even eliminate it. The best

production system which focuses on the elimination and achieves that successfully is the Toyota production system. And this represents the biggest difference between the Japanese production system and the European and American ones. While the Japanese system considers the inventory as an absolute evil, the others accept it as a necessary evil since it makes production runs smoothly. In this sense, the Japanese strategy is to avoid and remove all factors that necessitate the inventory [2]. One of the most crucial concept is cross-training. As much as the workers are cross-trained, the performance gets higher (of course, with considering the mechanism of applying). The cross-training offsets the benefits of inventory. Many initiatives have been adopted and effectively applied in this area example, cell manufacturing.

Recently, this idea has been changing a bit under the effects of many factors like the fast-changing products, small quantities with many types and so on. This requests a huge amount of restless training that consuming the time and money. As a result, many Japanese pioneers in cell manufacturing have started to accept a small amount of inventory.

The current concept is to mix between the cross-training and the capacity of buffer and try to make a best combination of both. It becomes reasonable to find a manufacturing cell with limited size of buffer.

2.2 Cell Manufacturing and DLB

Three types of manufacturing cells are famous in literature and application. These cells are single-worker cell or yatai cell, divisional cell and rotating [3], [4], [5], [6]. These cells are different in term of the range of tasks that each worker can do, the mechanism of their movement, and the number of workers in each cell.

Since our focus is on the narrow range of cross-training with effective mechanism to manage it, we consider the divisional cell with helping zone. In the divisional cell, a determined number of operators perform, usually in a U-line, a specific number of operations according to their skills. In this sense, it is applied during the primary phases of the transition from the conveyor line to a cell. The total number of operations is divided into the number of operators according to their abilities. The workers usually shuttle among several workstations to complete all processes assigned to each of them. With helping zone, the cell has several overlapping stations between the neighboring workers' zones as plots in figure 2-1.

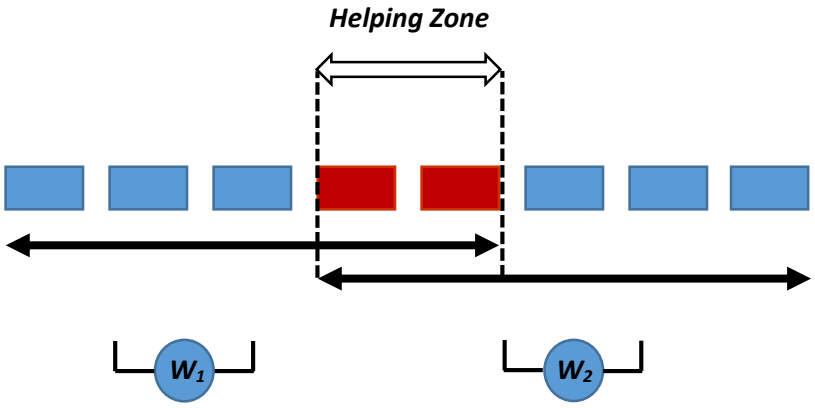


Figure 2-1 *Helping zone in the divisional cell*

DLB [7] has almost the same mechanism to do the job with one exception. That is no movement between stations each worker is assigned to one station as in a conventional serial production line. Several tasks that can be done at unique stations in the cell can be processed at one station in DLB line with ample necessary tools. The shared task in DLB corresponds to the helping zone in manufacturing cell. In this sense, the shared task or the helping zone might be attended by both adjacent workers at the same time or not in other words collaborative or non-collaborative [8].

2.3 Decision Points and Control Rule

How many subtasks the shared task can be divided into and are transferable is significant to ease managing the work sharing and increases the performance [9], [10]. Having more subtasks will reduce the idle time of next worker. Figure 2-2 demonstrates the idea of granularity of shared task. W_1 after he finishes his own fixed task should make a decision to send the shared task downstream or keep working on. The point at the beginning of each subtask of shared task is called decision point or *DP*. The size of shared subtasks between decision points represents by *n.*

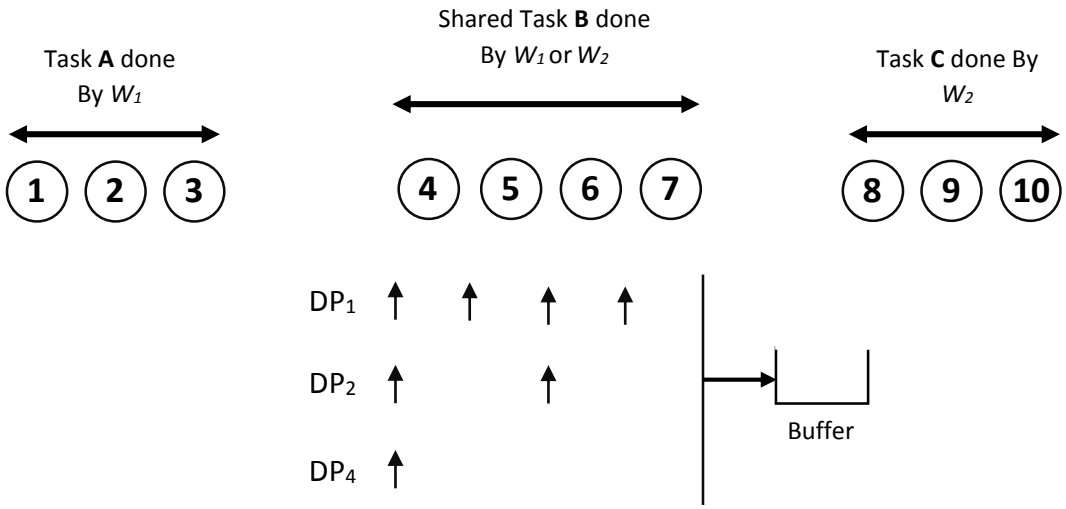


Figure 2-2 *Decision Points of Shared Task*

In the cell (we study here), W_1 would leave the job at some decision point inside the helping zone or at extreme at the end of helping zone. As the cell does not assume to have a buffer, W_1 keeps holding the job even if the shred task is finished. When W_2 become idle, the

job will be ready to transfer from W_1. Introducing the buffer will eliminate the blockage time. To control when the W_1 send the job to W_2 we employed the following rule displayed in the below flowchart figure 2-3.

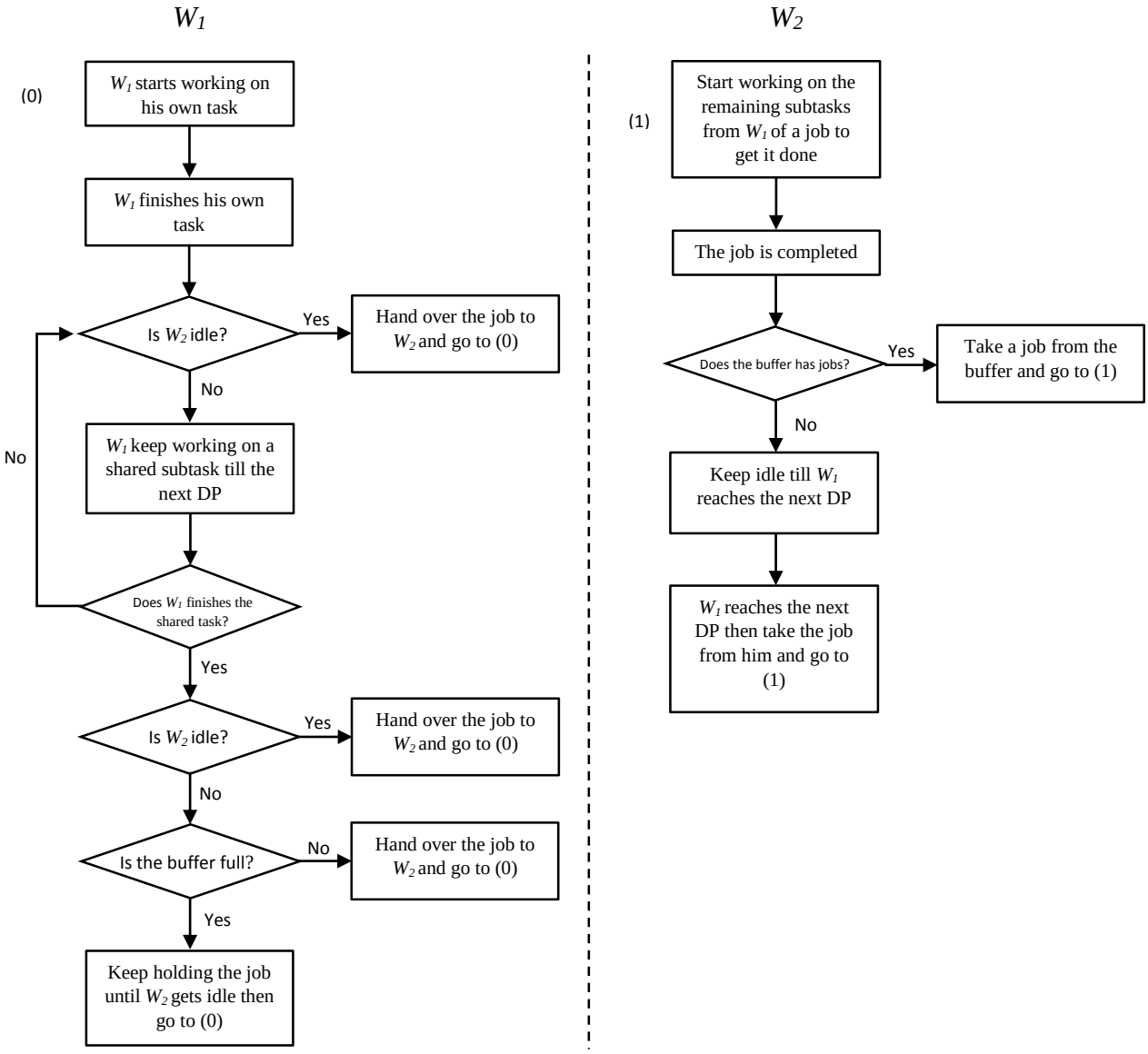


Figure 2-3 *Work mechanism of employed control rule*

The chart describes how the rule works from the viewpoints of both workers. When the W1 finishes his fixed task he checked if W_2 is idle or not. If he is idle the job transfer with shred

task undone. In case the W_2 is still busy, W_1 starts processing the shared task until he arrives to the next DP. He rechecks again the status of W_2 to send or go ahead to the next shared subtask and so on. Sometimes the W_1 finishes the whole shared task before W_2 gets idle. Then he send the job to the buffer between them. On the other hand, when W_2 finishes the job under his hand, he checks the upstream buffer and takes the job if any. If there is not job in the buffer he keep idle till W_1 finishes his fixed task or arrive to the next DP if he working on the shared task.

2.4 Settings of experiments

Simulation is done for a two-station line with two workers. The jobs are always available at the buffer before the first station. The time of job transmission between stations is negligible. The stations are considered close enough to neglect. The throughput **TH** is used to evaluate the performance.

The number of subtasks is 40 with 0.25 time unit for each subtask. The investigation is performed under the factorial experimental design of three levels for buffer size, three levels for the coefficient of variation of subtask time processing, five levels for the task division or the sizes of shared task (equal fixed tasks) and three levels of *DP*. The specific levels of each factor is as follows.

- Buffer (B): 0, 1, 2
- The number of subtasks of shared task: 0, 8, 16, 24, 32

  ie. The task division: 20-0-20, 16-8-16, 12-16-12, 8-24-8, 4-32-4
- DP Unit: DP_1, DP_2, DP_4

- Coefficient of variation 0.25, 0.5, 1.0

The experiments are conducted using the Visual SLAM (AweSIM ver. 3.02) [11] of 1.01 million time units for each experimental condition of a total of $3 \times 5 \times 3 \times 3 = 135$. The warm-up time is 10000 time units. Thus, the result is the same regardless of the initial conditions.

We used lognormal distribution to reflect the variability of processing time. In previous studies of DLB, exponential distribution is used mostly. Therefore, variation coefficient of processing time in that case is fixed at 1. To run experiments with other levels of the coefficient of variation (0.5 and 0.25), the lognormal distribution is a right choice. Comparing with Normal Distribution, the lognormal distribution avoids the negative values of processing time, as it comprises the right skirt of normal distribution (the positive values only as in the exponential distribution.

2.5 Results and Discussion

Table 2-1 shows the pooled analysis of variance, as TH is a respond. The analysis is under an error rate α less than 0.01 and high contribution pooling ratio R^2=0.996. From the table 2-1, it is clear that all factors are significant, as well as the two factors interactions and three factors interactions and the main ones.

Table 2-1 *Pooled analysis of Variance (Response is TH)*

Source	Degree of Freedom	Sum of Squares (10^{-6})	Mean Square (10^{-6})	F ratio	Contribution Ratio
Buffer	2	431.58	215.79	69076.0	.189
Shared Task	4	24.97	6.24	1997.8	.011
DP	2	488.09	244.05	78120.8	.214
Coefficient of Variability	2	504.46	252.22	80740.9	.221
Shared Task x Buffer	8	375.99	47.00	15044.7	.165
Shared Task x DP	8	128.79	16.10	5153.1	.057
Buffer x Coefficient of Variability	4	163.29	40.82	13067.3	.072
Shared Task x Coefficient of Variability	8	107.60	13.45	4305.5	.047
Buffer x Shared Task x Coefficient of Variability	16	44.69	2.79	894.1	.020
Error	80	9.65	0.12		.004
Total	134	2279.12			1.000

Figure 2-4 plots the main effects of studied factors. Increasing the buffer from 0 to 1 causes a big improvement in TH and after that it gets up slightly. The performance worsens, as the number of DPs gets small. In other words as many as the shared task get less granular the TH deteriorates. Unexpectedly, a line with no shared task or helping zone has the highest performance then TH decreases as the shared task gets bigger til 8. It goes up slightly after that as the size of shared task increases. For the variability, TH improves as the coefficient of variability get less.

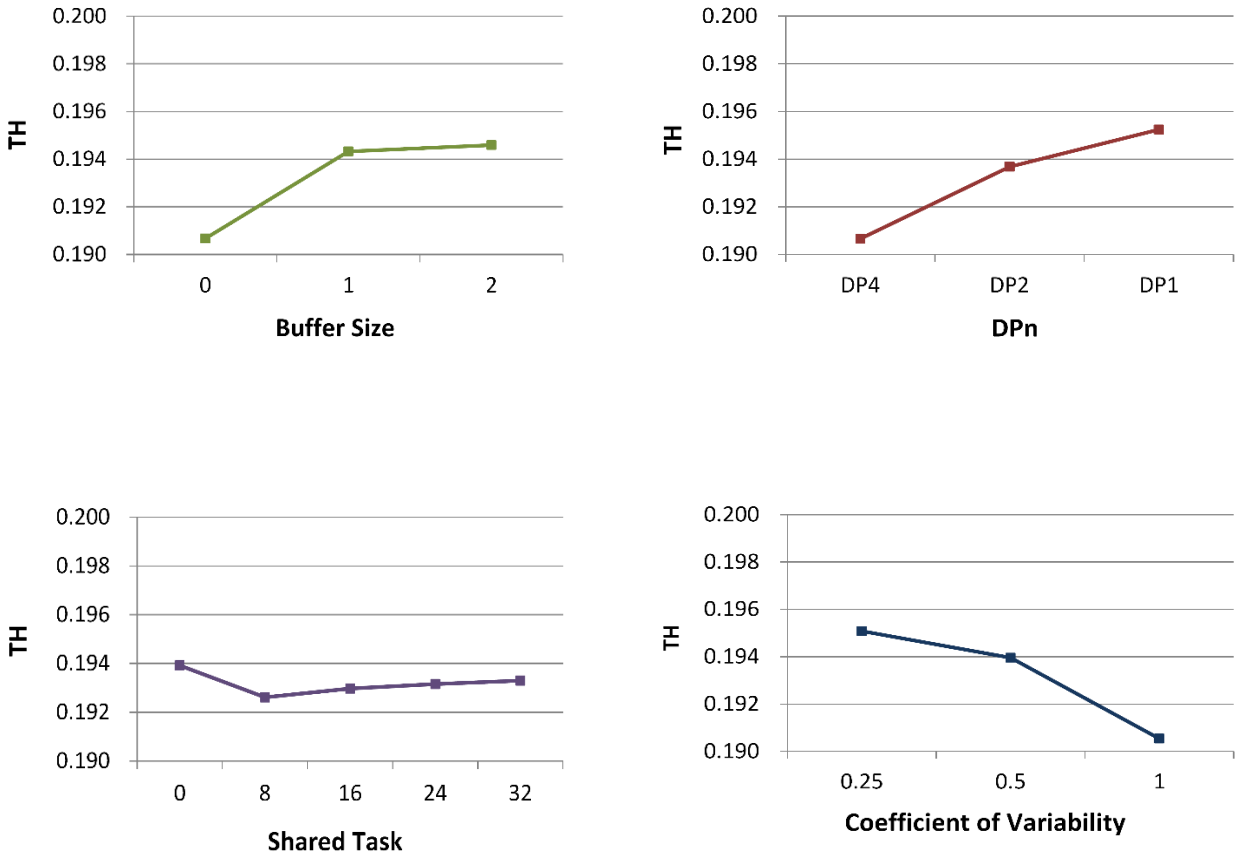


Figure 2-4 *The main effects of studied factors*

Now let's explore the performance for each combination of variability coefficient and DP through all levels of buffer size and shared task. Figure 2-5 depicts the results of TH through different combinations of studied factor levels. Each column contains the performance outcomes with the same variability coefficient and each line has the same number of DPs.

From figure 2-5 the following observations we can derive which basically depend on the variability coefficient.

1- When the coefficient of variability is small (0.25), there is an effect of the buffer only when the work-sharing is not applied. Expanding the buffer capacity by 1 or 2

shows a notable same improvement in TH. By introducing the work sharing, its effect can be noticed only when the shared task is so granular (DP_1) and with no buffer.

2- As the coefficient of variability raises, expanding the capacity of buffer with work sharing applied indicates a pronounced raise of TH. This improvement gets bigger as the number of DPs decreases. The improvement of TH due to the expansion of the scope of shared task becomes distinct even with a buffer of 1 only in one case. This case is when the variability is too high and the shared task is so granular (The number of DPs is big).

To conclude, the complementary effect of introducing the buffer on the performance of work sharing is not existed as long as the variability is small. On contrary. This effect is obvious when the variability is high and the number of DPs is small.

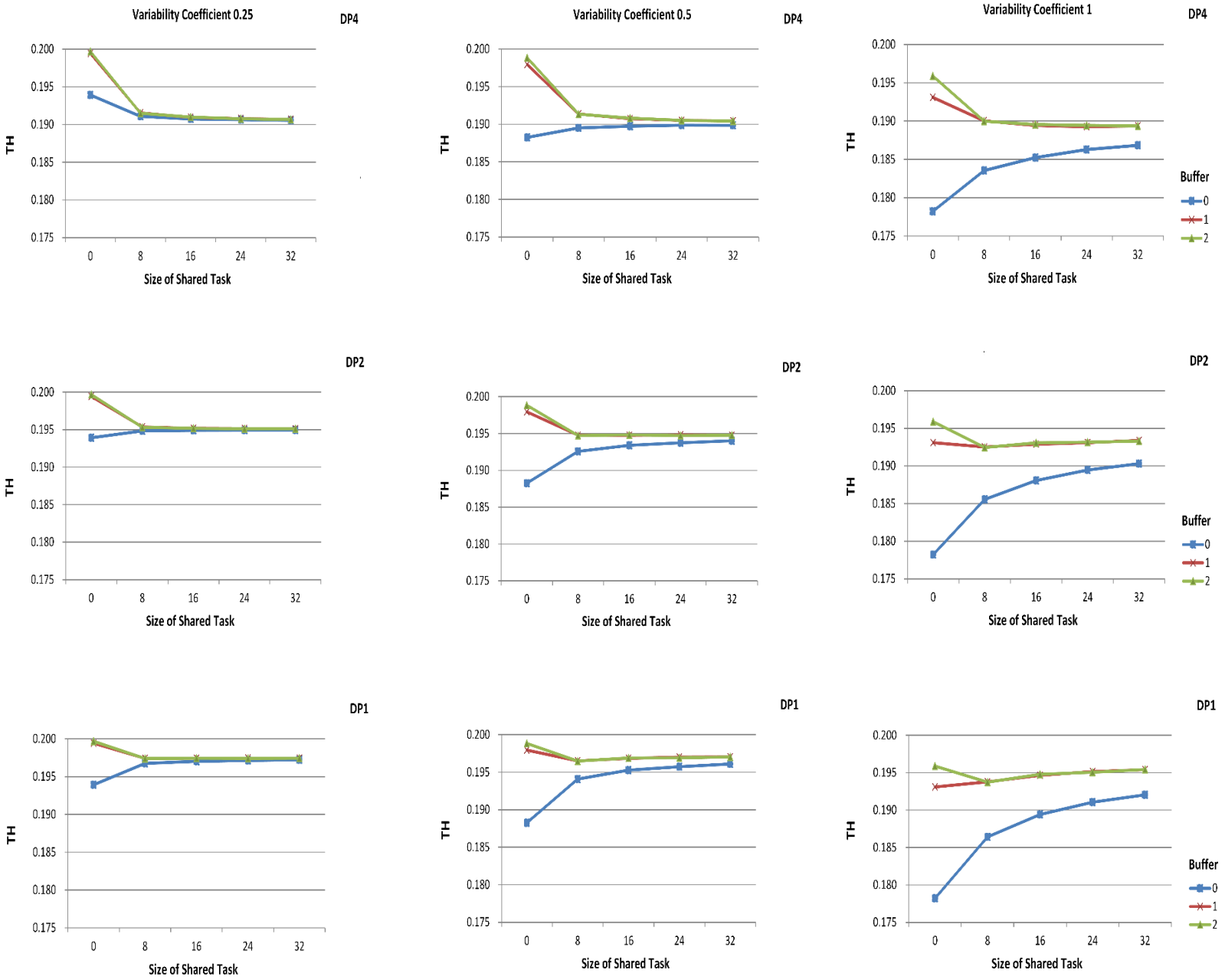


Figure 2-5 *The interaction effects of Buffer, coefficient of variability, size of shared task and DP*

Based on the results depicted in figure 2-5 (especially the left column), the following question arises. Why does the buffer has no a complementary effect on the performance of work sharing of scope (0-8) of shared task (helping zone) when the variability is small? And even its effect is negative. To address this point, we conducted experiments of the cases between (0-8) size of the shared task with DP1 and balanced fixed workload (the same settings of the leftmost chart of the last row of figure 4-5). Since we have already the results of the cases 20-0-20 and 16-8-16, the additional cases needed to experiment are 19-2-19, 18-4-18, and 17-6-17.

Figure 2-6 presents the average number of work-in-process in the buffer. As soon as the shared task (helping zone) is introduced as in figure 2-6 from 2 and so on, the buffer capacity is underutilized. Accordingly, the line with work sharing does not benefit from the advantage of expanded capacity of buffer.

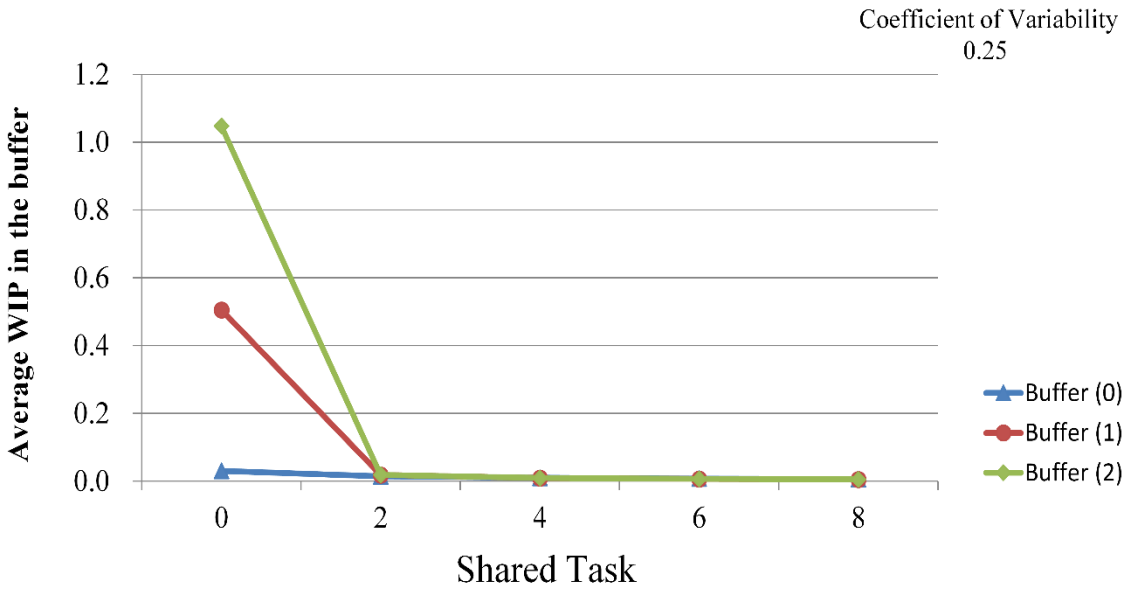


Figure 2-6 *The average work-in-progress in the buffer*

The reason behind this trend is that W_2 has to wait when W_1 is still working on the shared task till next DP to receive the job when it gets idle. If the shared task has one DP_1, W_2 will experience longer waiting time than the case with more DPs. The line without work sharing does not suffer from this issue. In this line, W_1 will drop the job as soon as he or she finishes his own job. As the variability increases this scenario with work sharing happens in less frequency. With high variability, the situation is reversed and the complementary effect of buffer become remarkable.

In the above analysis, the fixed workload assigned to each worker is equal. In this case, how many DPs the shared task has plays the important role to reduce the long waiting time W_2 would experience. However, having many DP of shared task is sometimes unrealistic or undesirable for technical or quality related issues. As an another option, what if W_2 has a bigger fixed workload from the beginning. That might shorten the idling time of W_2. To investigate this point, we simulated a line with DP_1 and coefficient of variability equal to 1. Four task divisions are simulated, which are 14-8-18, 10-16-14, 6-24-10, 2-32-16. Table 2-2 displays the results as the buffer capacity grows.

Table 2-2 *TH vs. Buffer Capacity with overloaded station 2*

Task division		20-0-20	14-8-18	10-16-14	6-24-10	2-32-6
Buffer Capacity	0	0.1782	0.1856	0.1892	0.1907	0.1917
	1	0.1931	0.1968	0.1959	0.1959	0.1957
	2	0.1959	0.1970	0.196	0.1957	0.1957

It is obvious form the table 2-2, the imbalance workload with more fixed subtasks for W_2 support the trend to introducing the buffer between stations. The complementary effect of

buffer under work sharing applied becomes remarkably important. The performance is better even than the balanced case. For example, TH of 14-8-18 for buffer capacity of 1and 2 are 0.1968, 0.1970 respectively. This performance is better than the case with same shared task and balanced workload (16-8-16) where TH is .01937 for buffer capacity of 1, 2.

We have employed so far a standard rule to manage the work sharing. In this rule, the status of W_2 is considered to keep or send the job with undone shared task. In DLB, many rules are investigated. HFB (Half-full- Buffer) is the most excellent one among them. In this rule, the amount of available downstream work is utilized to check the status of W_2. The available downstream work includes the remaining subtasks unfinished at station 2 and the number of subtasks in the upstream buffer. This amount is compared with the cutoff value called *R*. if the available downstream work is bigger (smaller) than *R*, W_1 keep the job to do the shared task till next DP and keep check till next other DP and so on till the shared task is done (send the job without shared task done). The *R* in this rule is given as follows

$$R = \frac{t_Y}{2} + \frac{(t_Y + t_Z) + t_Z}{2} \cdot \frac{B}{2} \qquad (1)$$

Where

t_Z the number of subtasks of the shared task

t_Y the number of subtask assigned to W_2

B the capacity of buffer

We did the experiments with DP_1 and the coefficient of variability of 1. Figure 2-7 shows the performance (TH) of the two rules; the standard (solid line) and HFB (dotted line) rules.

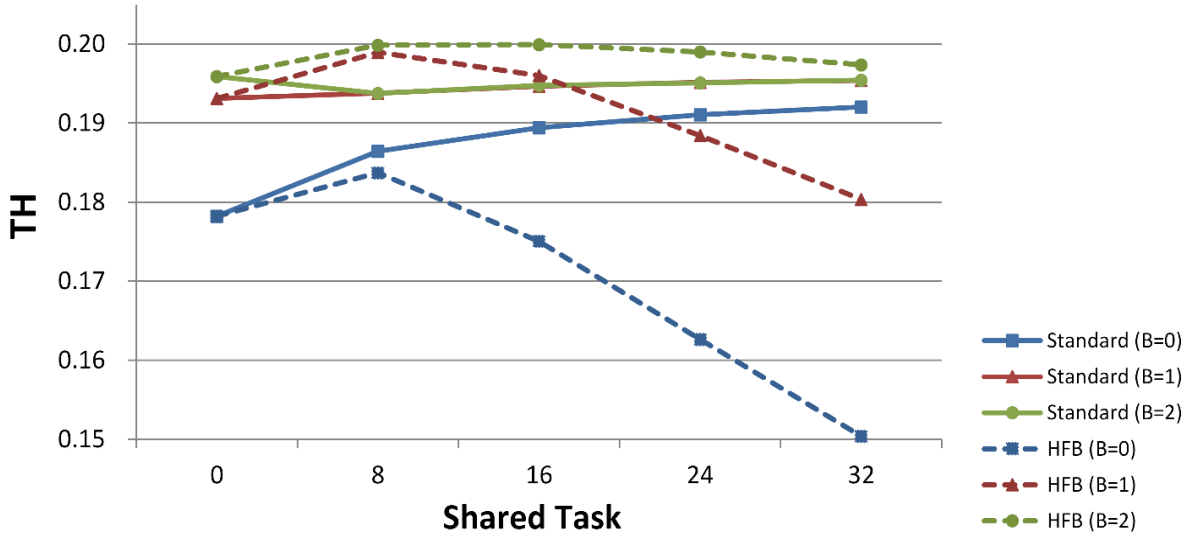


Figure 2-7 *The performance of HFB and Standard rules*

The performance of HFB get high as much as the buffer capacity increases for the same task division. On the opposite side, TH deteriorates for the same capacity of buffer as the shared task or helping zone expands. The performance decline gets severe when the line does not have a buffer.

From above, we can come to the following conclusion. In HFB, as the helping zone size increases, the effect of buffer becomes much significant on the performance than the standard rule. In addition, the big deficiency of HFB is the need for high level of information about the work available at station 2 and in the buffer. That is not easily practical in the reality.

2.6 Conclusion

The effect of introducing a buffer in the cell manufacturing represented by a divisional

cell was addressed. The cell with helping zone is simulated using DLB framework to investigate the complementary effect of buffer.

Two cases depending on the coefficient of variability have arisen. When the coefficient of variability is small, the buffer effect is not observed where the work sharing is applied. To benefit from introducing the buffer, either the line should be running without work sharing or making the fixed workload of station 2 is bigger than the preceding station. The other case is the coefficient of variability is large. The buffer capacity of 1 is enough to improve the performance and increasing the buffer to 2 does not raises the value of TH. The effect of buffer with applying the work sharing becomes much clearer as the number of DP decreases.

In cell manufacturing, a line with cross-trained workers and low variability does not need a buffer and the work sharing can absorb the variation. On opposite case, if the variation is high or the skills of worker are not sufficient, the introduction of buffer can help remarkably improving the performance.

As using the robots in the cell increases, control rules like HFB that requires a high level of information accuracy could be a good choice. That urges for additional investigations to find out the cases requiring the buffer or not to improve the performance under such a rule.

References

[1] Conway, R., Maxwell, W., McClain, J.O., and Thomas, L.J., 1988. The role of work in process inventory in serial production lines. Operations Research 36 (2): 229–241.

[2] Shingo, S., 1985. Non-stock production: the Shingo system for continuous improvement. Productive Press. xx.

[3] Nakade, K., and Ohno, K., 2003. Separate and carousel type allocations of workers in a U-shaped production line. European Journal of operational Research. 145: 403-424.

[4] Miltenburg, J., 2001. U-shaped production lines: A review of theory and practice. International Journal of production Economics. 70: 201-214.

[5] Isa, K., and Tsuru, T., 1999. Cell production and workplace innovation in Japan: Toward a new model for Japanese Manufacturing. The Institute of Economics Research Hitotsubashi University, Discussion Paper Series A. 360.

[6] Yin, Y., Kaku, I., Murase, Y.,and Yasuda,Y., 2008, Seru (Cell) and Reverse Conversion: Part I. Definition, Self-evolution, and Typology, Research Group of Economics and Management, No. 2008-E01.

[7] Ostolaza, J., McClain, J. and Thomas, J., 1990. The use of dynamic (state-dependent) assembly-line balancing to improve throughput. Journal of Manufacturing and Operation Management, 3:105-133.

[8] Van Oyen, M.P., Gel, E.S., Hopp, W.J., 2001. Performance opportunity for workforce agility in collaborative and noncollaborative work systems. IIE Transactions 33, 761–777.

[9] Gel, E. S., Hopp, W. J. and Van Oyen, M. P., 2002. Factors affecting opportunity of worksharing as a dynamic line balancing Mechanism. IIE Transactions, 34(10): 847-863.

[10] Chen, J., and Askin, R.G., 2006. Throughput Maximization in Serial Production Lines with Work-sharing. International Journal of Production Economics, 99:88–101.

[11] Alan, A., Pritsker, B., and O'Reilly, J. J., 1999. Simulation with visual SLAM and AweSim. 2nd Edition, John Wiley & Sons, New York.

CHAPTER 3

DYNAMIC LINE BALANCING (DLB) AND FIXED WORKLOAD

Overview

This chapter explains the concept of Balancing Line Balancing DLB and the coordinate rules utilized to manage the work sharing. It also illustrates the model we employed in this research and the workload measure we made up. The results here show the effect of workload on the performance under this basic model at this stage.

3.1 Introduction

The work-sharing in the serial lines is distinguished from the one in non-serial lines (cells). In most cases, the cross-trained worker can do the task that is directly coming after his own task or/ and the one directly preceding his own task.

From the work-sharing viewpoint, a serial line has two levels of work sharing, no work sharing, partially work sharing. For no work sharing, worker 1 sends a job immediately after he finishes processing. While in partially work sharing, he might keep the job to do the next task, as the next worker is still busy.

To simplify the idea, suppose a job needs three consecutive tasks to be completed A, B, C by two workers. With no work sharing (Figure 3-1), these tasks should be assigned to the workers in the following two possibilities; AB-C, A-BC. That is, the task B is assigned to one

of these two workers permanently. In this case, worker 2 has to wait until worker 1 finishes all his processing to start working if the number of jobs in the line not enough or it should support the buffer between the workers with enough semi ready jobs (in these jobs the task of worker 1 is done). That can be done by letting worker 1 starts processing earlier than worker 2 with enough interval or getting a support by adding an additional worker. These countermeasures can be applied if the worker 1 is a bottleneck. On the other side, if worker 2 is a bottleneck, inserting more jobs in the line to provide the worker 1 with enough work might be the correct action. In all above scenarios, the line has to have extra jobs to attain the targeted throughput and as a result, many jobs will wait which, prolongs the cycle time. Moreover, quality problems are easy to be hidden and difficult to reveal in such lines.

(a) (b)

Figure 3-1 *Tasks assignment with no work sharing*

Figure 3-2 *Tasks assignment with partially work sharing*

With partially work sharing (Figure 3-2), both adjacent workers can do the shared task. Worker 1 might keep the shared task after he finishes his own task to process it or send it to the

downstream buffer. That depends on the status of the downstream (the mechanism to control this decision is explained in modeling background). This approach minimizes the probability of having the worker idle and reduces the need for more jobs to achieve the targeted throughput. Such kind of this line is easier to control and discover the quality matters.

3.2 Dynamic Line Balancing

Dynamic Line Balancing DLB policy, introduced by [1], is one of the famous and efficient ways to apply the work sharing. In this technique, the worker will not move between stations as for example in the Bucket-Bridge System [2]. Each worker stays at his station and does the shared work. Each job has two types of tasks, fixed tasks which can be done only by a designated worker. The other type is called shared tasks that can be done by either of an adjacent pair of workers. A worker chooses to pass on a job with the shared task undone or complete the shared task depending on the system status.

The purpose of DLB is to maximize the efficiency by balancing the line. As much as the line approaches the balanced case, its efficiency is being close to the ultimate highest performance. It might be asked that it could be achieved by balancing the mean processing times that can be referred to as static balancing. Actually, the manufacturing process exposes many sources of noise and some of them are difficult or not cost-effective to manage. The effect of those sources could be represented as variability. As results, the line becomes unbalanced in the short term. In this sense, on-the-fly or state-dependent balancing (dynamic balancing) is more effective than static.

3.3 Model Description

We use a two-station production line. One worker is at each station. W_1 attends at station 1 and W_2 attends at station *2*. The movement between stations is not allowed. The buffer capacity is infinite, but the total number of jobs in the line is restricted to *WIP* number of jobs. This inventory level is kept constant by the CONWIP (CONstant-Work-In-Process) policy. CONWIP, introduced by [7], [8], keeps Max*WIP* constant by preventing a new job to enter the line until a finished job leaves when the number of jobs reaches to Max *WIP* level. This policy eases controlling the number of jobs in the line, which affects the performance whether work sharing is used, or not. Figure 3-3 illustrates how CONWIP policy works comparing with others control policies.

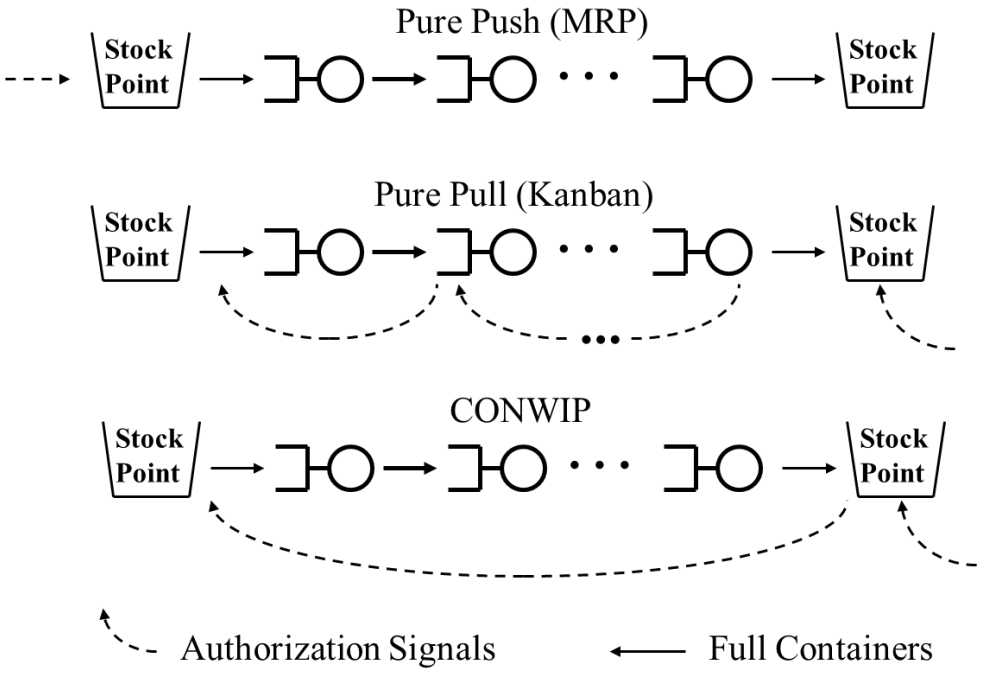


Figure 3-3 *CONWIP, Pure Push, Pure Pull*
Source: [9] Hopp and Spearman, 2008 "Factory Physics"

Having a high level of *WIP* in the line normally results to improve the throughput. Since the goal is to explore the work-sharing and the high level of WIP in the line will hide the work-sharing influence and prolong the cycle time and other side consequences accompanying that. The study is done with small numbers of *WIP*; 3, 4, 5. Bokhost [3] demonstrated the bad effects of having large amount of WIP on the balanced use of cross-trained skills. With the ample WIP, the workers will work more on the familiar tasks and hardly use and maintain the newly acquired skills. He ended that reducing the amount of work in process forces the workers to make a more balanced use of skills they have.

Basically, in DLB each job consists of two types of tasks, the first type is unique tasks for each worker and the other tasks are shared tasks between each neighboring workers. For example in a model with two stations as in our model, there are task *A*, task *B*, and task *C* done in this order. Task *A* can only be performed by W_1 at station1 and task *C* can only be performed by W_2 at station 2. Both of tasks *A*, *C* are called fixed tasks. Task *B* can be done by W_1 or W_2 at his own station and it is called shared task. We assume there are enough equipment and tools for both workers to avoid any waiting for occupied tools available at their stations to operate task *B*. we consider the shared task *B* is non-preemptive task. When a worker starts processing the shared task, it can be released until it becomes completed.

The tasks that forms a job is modelled following the subtask model introduced by [10]. In this model, the job consists of *T* numbers of equal subtasks to be performed in sequence. Based on this model, we represent the processing time of each task (*A*, *B* and *C*) by the number of subtasks where each subtask has a processing time of one unit time and the subtasks are identically distributed. For example, in t_A- t_B - t_C task division, task *A* has t_A subtasks, task *B* has t_B subtasks, and task *C* has t_C subtasks. Dividing the job's tasks into a number of equal subtasks

has three advantages (as mentioned in [10]). Firstly, the variability of the total processing time of completing a job does not change with different configurations of task division. Secondly, it can represent most production systems where tasks are grouped to help balance the line. Finally, the variability of processing time is better to be modelled by the subtask model.

The job is processed in first come first serve (*FCFS*) queue. The job goes in one direction from upstream to downstream stations. If a job is sent to the downstream station, it cannot return to the previous station. The workers are equal in speed and W_1 is the one who decides to pass on or keep working on the shared task.

3.4 Coordinate Rules

In research model, both workers can do the same shared task (but not at the same time). Having small *WIP* in the line is one of the most important advantages to invest in work-sharing. In this sense, if worker 1 (W_1) does the shared task most of the time that leads to worker 2 (W_2)' starvation and if W_2 does it most of the time W_1 might starve for long time. As W_1 who the only can make the decision to send or keep the job, he / she has to consider the available work downstream. After completing his fixed tasks, W_1 should push a job with uncompleted shared task when he finds out that W_2 might become idle in the near future while he is busy or at least reducing the probability of W_2 being idle.

The rules and policies that describe the mechanism of making the decision of doing the shared task or not, should have three characteristics. These characteristics can be similar but still with some differences from that in [3]'s paper. He labels them as three rules to assign the work in DRC (Dual resource constrained) studies. The three features are

(1) at what moment the job can be sent downstream,

(2) which job should be processed first.

(3) The last is which worker should process the shared task when there are more than one skilled worker can do it.

In DLB environment, the first characteristic has three options which are (1) after completing the whole shared task (non-preemptive shared task), (2) at natural breaking points (as with granularity mentioned in [4]), and (3) at any moment (preemptive shared task). The second characteristic has two popular methods in DLB which are FCFS (First-Come-First-Serve) or STP (i.e. Do the shortest job from the upstream buffer) order [1]. The final characteristic is what is usually called in DLB, decision or control rules.

Two kinds of decision rules can be found in DLB literature. The first one is the optimal control rule. This rule is the solution of optimization problem of stochastic model that describes all the system states. The system states increase as *WIP* increases. For example, a model of 10 WIP has 1023 states for which worker needs to know the proper action (see [4]). Moreover, the optimal rules do not have a simple structure, which makes it easy to apply by workers in the line. In the light of these disadvantages, the second type of rules is the threshold rules are more practical in the real production environment. Those rules are not generally optimal. However, there are several of them are near optimal with a small accepted level of deviation from the optimal situation. In the threshold rules, after W_1 finishes the fixed tasks, the decision is made based on the comparison between the available downstream workload and a specific threshold called *R*. Many threshold rules are found in the literature. These rules can be distinguished based on the information complexity level required (which are mentioned before).

Among of threshold rules from these two groups, we used one rule from each class that is the most near-optimum based on [5]. Of the first group, SRNS is used and of the second HFB is used. In SRNS (HFB) (see [5],[6]), W_1 starts working on the shared task if R or more units of subtasks are available in the buffer before (and at) station 2. And W_1 will pass the shared task if the subtasks in buffer before (and at) station 2 are less than R. The threshold value R is given by equation (1) for SRNS and equation (2) for *HFB*, where *WIP* is the CONWIP level and t_C (t_B) is the number of subtasks for task C (B):

$$R = \begin{cases} (WIP-2)t_C, & if\ t_B \geq (WIP-2)t_C, \\ [t_B\ , (WIP-2)t_C] & if\ t_B < (WIP-2)t_C \end{cases} \quad (1)$$

$$R = \frac{t_B}{2} + \frac{(t_B + t_C) + t_C}{2} \cdot \frac{WIP-2}{2} \quad (2)$$

3.5 Workload Measure

In DLB research, the workload and its distribution are rarely treated. Even when it is tackled, it is with some examples (as in [4], [5]), or mentioned briefly (as in [4]). The balanced fixed workload with different sizes of the shared task is the most common case in the research. However, unbalanced fixed workload might be more realistic. The unbalance could come from the nature of processing that does not allow dividing the workload to equal fixed tasks because of technical or quality issues.

The shared task can be managed by the decision rules and of course can help alleviating the unbalanced workload (see [6]). On the other hand, having a clear full picture of workload effect can help the production stuff to find ways to manage this workload by for example altering the order of processing some fixed task from W_1 to W_2 which might lead to higher

efficiency. For that reason and others, we simulate variety of fixed workload configurations to clarify the workload effect and its trends.

From this point, workload refers to the size of fixed tasks and their distribution between the stations. For example, *4-2-4* task division has equal fixed tasks while *6-2-2* has different fixed task. *2-2-6* task division has the same shared task's size but the distribution of the fixed workload is opposite to the former division.

In this research, a new way to measure the workload is proposed to ease investigating and comparing the different task divisions with different workload configurations. The measure of workload starts from a theoretical assumption that all task divisions can be returned to the case of balanced line with only two equal tasks. Then, the shared task is composed from some last work units of station *i* and some first work units of station *j*.

Let 'suppose a job with fixed tasks and shared task that is shown in figure 3-4, the virtual breakeven point BP divides the job's work units into two equal groups of work units *L*. The shared task consists of some work units from station *i*'s side and some work units from the other side *j*. ST_{ij} *(*ST_{ji}*)* represents the size of shared task that can be done by worker *i (j)* from station *j (i)*'s side.

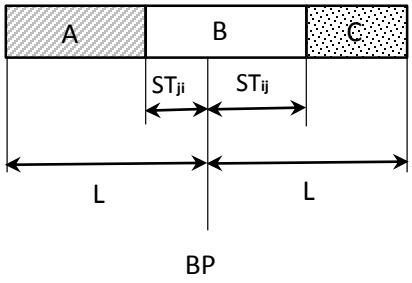


Figure 3-4 *The illustration of the proposed measure for the workload distribution*

The measure of workload α_{ij} is defined by the ratio of the shared work units' number done by worker *(i)* from station *(j)* side to the total shared task's size as if all subtasks are distributed equally between two adjacent workers. The measure is correct as long as both of t_A and t_C are less or equal to L. The measure is given as follows;

$$\alpha_{ij} = ST_{ij} / B \tag{3}$$

$$B = ST_{ij} + ST_{ji} \tag{4}$$

With a numerical instance, let's have a job with *4-4-2* task division. We have B=4, ST_{12} =3, ST_{21} =1 and α_{12} =3/4, α_{21}=1/4. Task division *5-3-2* has B=3, ST_{12} =3, ST_{21} =0 and α_{12} =1, α_{21}=0. To explore several cases and get the more reliable general results, ST_{21}, ST_{12} each are given four values; 1, 2, 3, and 4. Therefore, we get 24 cases. Table 3-1 presents some of these cases with calculated α_{12}, α_{21} based on equation (3).

Table 3-1 *The studied workload configurations*

A	B	C	ST_{12}	ST_{21}	α_{12}	α_{21}
5	0	5	0	0	-	-
4	1	5	0	1	0	1
3	2	5	0	2	0	1
2	3	5	0	3	0	1
1	4	5	0	4	0	1
5	1	4	1	0	1	0
4	2	4	1	1	0.5	0.5
3	3	4	1	2	0.33	0.67
2	4	4	1	3	0.25	0.75
1	5	4	1	4	0.2	0.8
5	2	3	2	0	1	0
4	3	3	2	1	0.67	0.33
3	4	3	2	2	0.5	0.5
2	5	3	2	3	0.4	0.6
1	6	3	2	4	0.33	0.67
5	3	2	3	0	1	0
4	4	2	3	1	0.75	0.25
3	5	2	3	2	0.6	0.4
2	6	2	3	3	0.5	0.5
1	7	2	3	4	0.4286	0.5714
5	4	1	4	0	1	0
4	5	1	4	1	0.8	0.2
3	6	1	4	2	0.67	0.33
2	7	1	4	3	0.5714	0.4286
1	8	1	4	4	0.5	0.5

3.6 Settings of Experiments

Visual Slam language (A simulation language) through AweSim software [11] is used to model and execute the simulation. The workers are working for 8 hours per day. The simulation is run for one year for each configuration (1year * 250 days * 8 hours * 60 min = 120,000 min) with four replications and the warm-up period is 10000 min.

The processing time of a subtask is exponentially distributed with mean of one, and a task is Erlang-k where k is equal to the number of subtasks composing this task. The total processing time of the job is 10 minutes plus the variability. The control rule used here is SRNS rule and we will present the results with the HFB rule in the next chapter in the context of comparison with SRNS rule.

To evaluate the performance, the efficiency is considered as a measure to evaluate the performance. It is defined as the ratio of simulated throughput rate TH^s over maximum achievable throughput rate TH^* with balanced line, and deterministic processing times. This measure is given by Equation (5);

$$E = \frac{TH^s}{TH^*} \tag{5}$$

3.7 Results and Discussion

The cases of study are combined by using the average efficiency into five combinations. Each combination represents a range of α values as shown in Table 3-2. The average results of efficiency of these combinations show three distinguished patterns according to *WIP* level.

Table 3-2 *The five combinations of workload measure α*

Combination	$\alpha_{12} \in$	$\alpha_{21} \in$
$\alpha_{12} = 0 , \alpha_{21} = 1$	[0,0.25[	]0.75,1]
$\alpha_{12} = 0.25 , \alpha_{21} = 0.75$	[0.25,0.5[	]0.5,0.75]
$\alpha_{12} = \alpha_{21} = 0.5$	0.5	0.5
$\alpha_{12} = 0.75 , \alpha_{21} = 0.25$	]0.5,0.75]	[0.25,0.5[
$\alpha_{12} = 1 , \alpha_{21} = 0$	]0.75,1]	[0,0.25[

The first, when *WIP*=3. Figure 3-5 presents the results. The plot shows that as α_{12} increases the efficiency increases till it reaches the value 0.75 then decreases. α_{21} has an inversed pattern since $\alpha_{12}+ \alpha_{21} =1$. Where the performance increases till 0.25 then starts decreasing. The highest efficiency is achieved for (α_{12}=0.75, α_{21}=0.25) while the lowest efficiency is achieved for (α_{12}=0, α_{21}=1). For example, 4-4-2 has higher efficiency than 1-4-5.

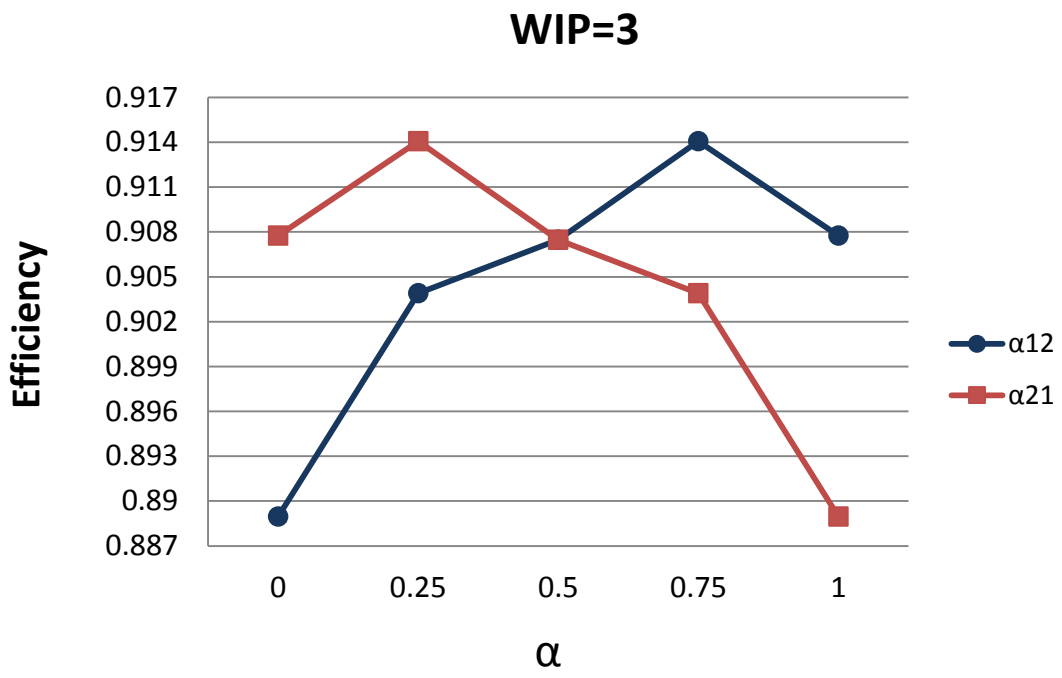


Figure 3-5 *The change of efficiency with α under WIP=3*

By comparing between (α_{12}=0, α_{21}=1), (α_{12}=1, α_{21}=0), we find that the second combination presents higher efficiency. The same is for (α_{12}=0.25, α_{21}=0.75), (α_{12}=0.75, α_{21}=0.25), but with better performance than the previous. The reason might be that when α_{12} is 0 or small, it means a large fixed task for station 2. As a result, this large fixed task will cause a smaller throughput since station 2 becomes a bottleneck. This could be alleviated as the fixed task of station 2 get smaller or in other word as α_{12} becomes bigger. However, this amount of fixed task or α_{12} has a limit which is 0.75 after that the effect gets inversed as station 1 will be

the bottleneck. Beside to the above reason, the small number of WIP in the line deepens this difference, since the station one could starve. Another observation is when $\alpha_{12}=\alpha_{21}=0.5$, the efficiency is between ($\alpha_{12}=0.25$, $\alpha_{21}=0.75$), ($\alpha_{12}=0.75$, $\alpha_{21}=0.25$). Let us take an example to make the previous discussion clearer. Figure 3-6 represents the different combinations of α under $B=3$.

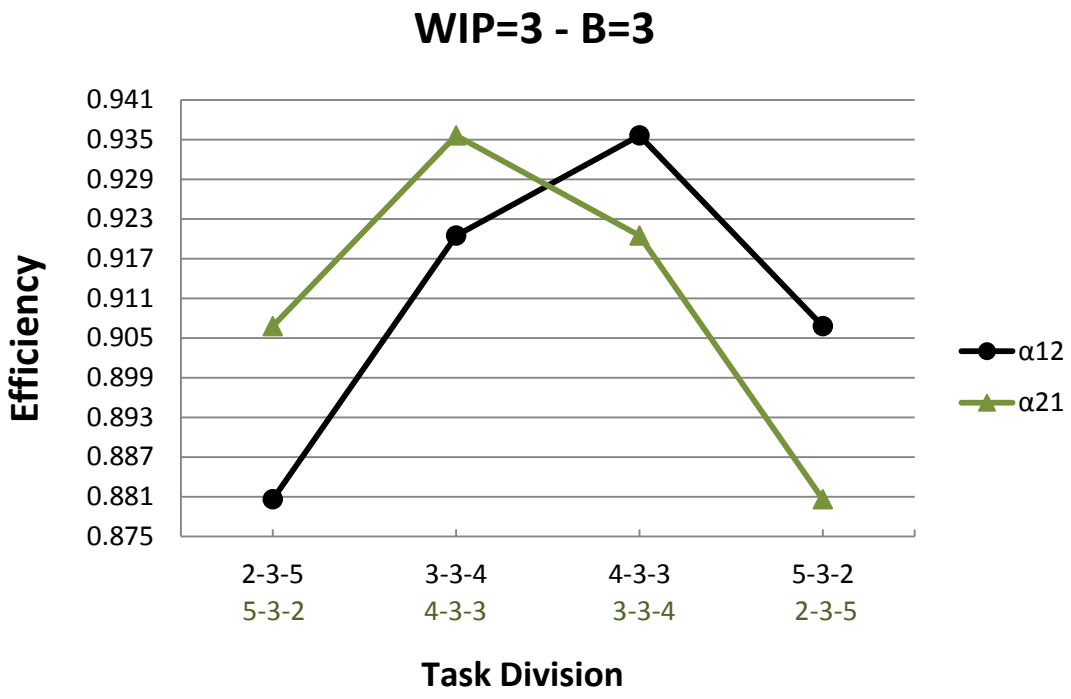


Figure 3-6 *Efficiency with different task divisions and B=3*

The second case when *WIP*=4. Here as in Figure 3-7, the differences between the opposite combinations get smaller. ($\alpha_{12}=0.25$, $\alpha_{21}=0.75$), ($\alpha_{12}=0.75$, $\alpha_{21}=0.25$) have small difference and also for ($\alpha_{12}=0$, $\alpha_{21}=1$), ($\alpha_{12}=1$, $\alpha_{21}=0$). However, the other results are same except for $\alpha_{12}=\alpha_{21}=0.5$ which becomes as same as ($\alpha_{12}=0.75$, $\alpha_{21}=0.25$). Here, the surplus of WIP in the line moderates the starvation of station one, which results to minimize the differences between the opposite combinations.

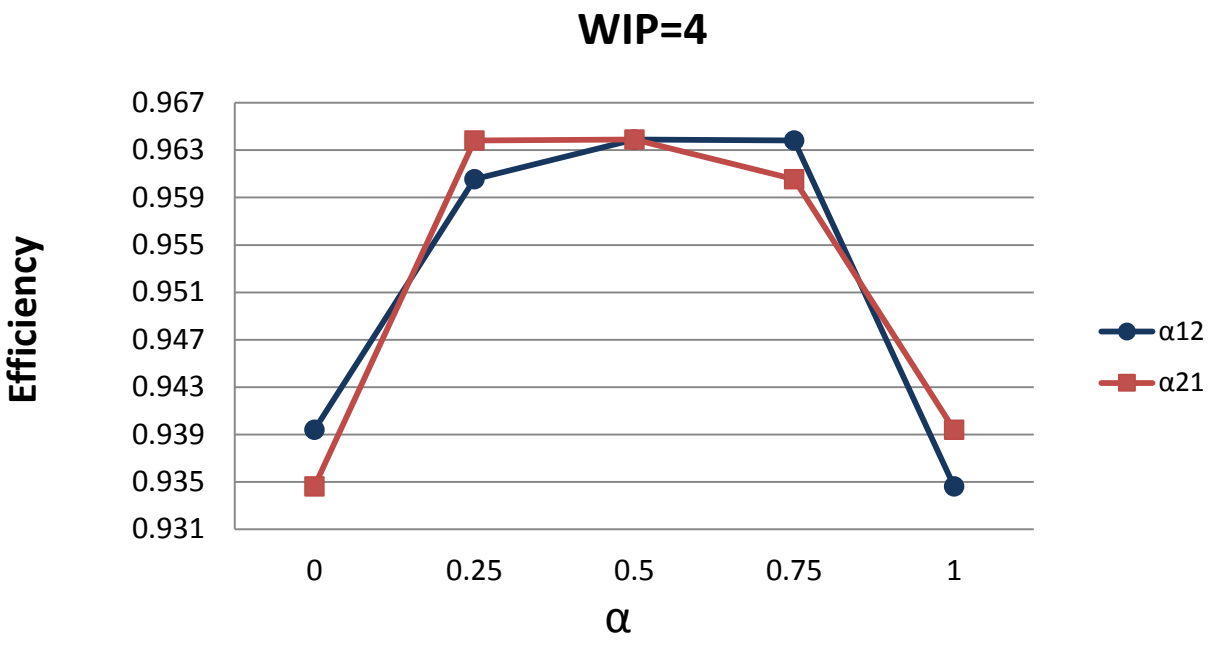


Figure 3-7 *The change of efficiency with α under WIP=4*

For example, with B=4, we get Figure 3-8 which demonstrates the pattern of workload when *WIP* increases to 4.

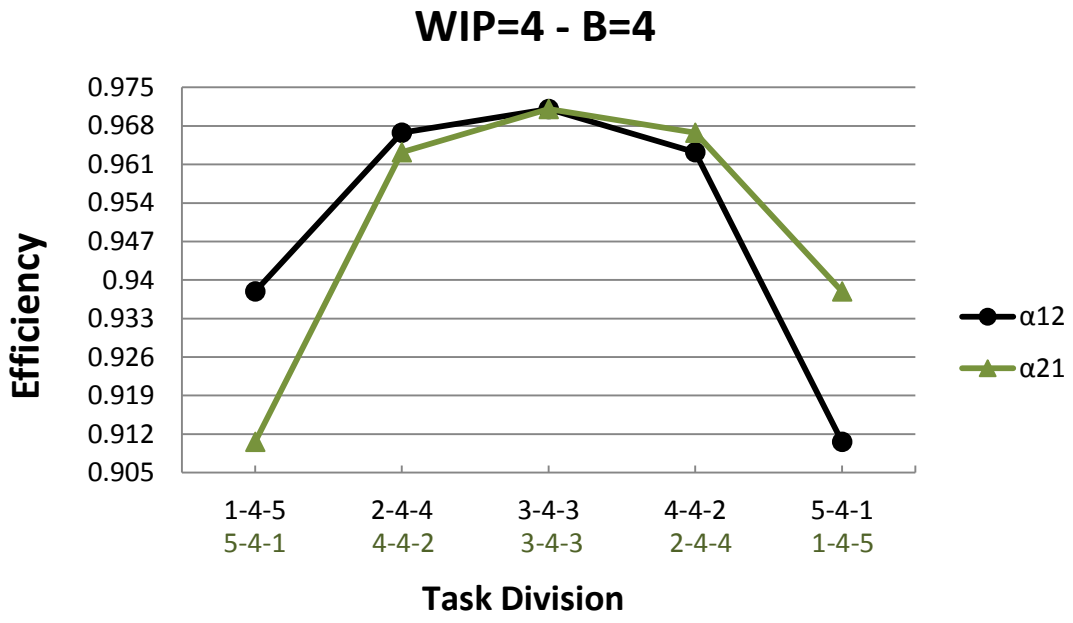


Figure 3-8 *Efficiency with different task divisions and B=4*

The last case when *WIP* equals to five. The pattern is opposite to the first case and the differences between the opposite combinations are less. As in figure 3-9, the efficiency get better as α_{12} get bigger then decreases after 0.5 while the efficiency of α_{21} continues improving till 0.75. The combination (α_{12}=0, α_{21}=1) shows higher efficiency than (α_{12}=1, α_{21}=0). Also the combination (α_{12}=0.25, α_{21}=0.75) has better efficiency than (α_{12}=0.75, α_{21}=0.25) which is opposite to the case with *WIP*=3. (α_{12}=0.25, α_{21}=0.75) gives the highest efficiency and α_{12}=α_{21}=0. 5 also has the same efficiency.

When α_{12}=1, that means the fixed task of station 1 gets so big (in this study 5), in other words it becomes a bottleneck. That will prevent inserting more jobs into the line resulting starvation in the station 2. On the other hand, when α_{12}=0, which means the station 2 gets a bottleneck. Here, no station will starve since there is a surplus of *WIP*. For α_{12}=0.25 and 0.75, the same analysis can be considered. As long as α_{12} is between 0 and 1, that will result better performance since the effect of bottleneck will be less. Figure 3-10 plots the same pattern as discussed before with *B*=5.

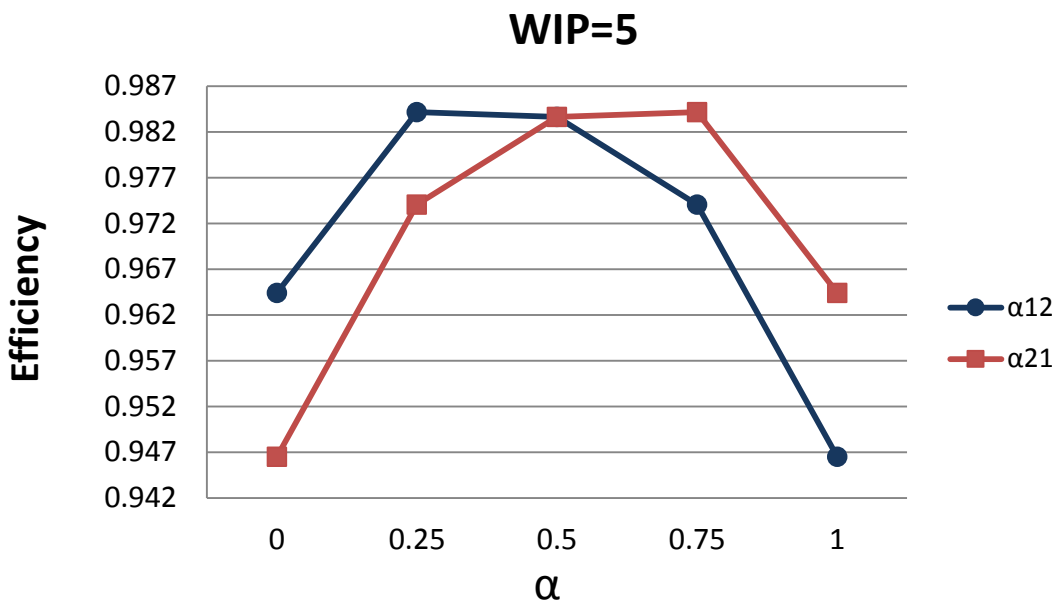


Figure 3-9 *The change of efficiency with α under WIP=5*

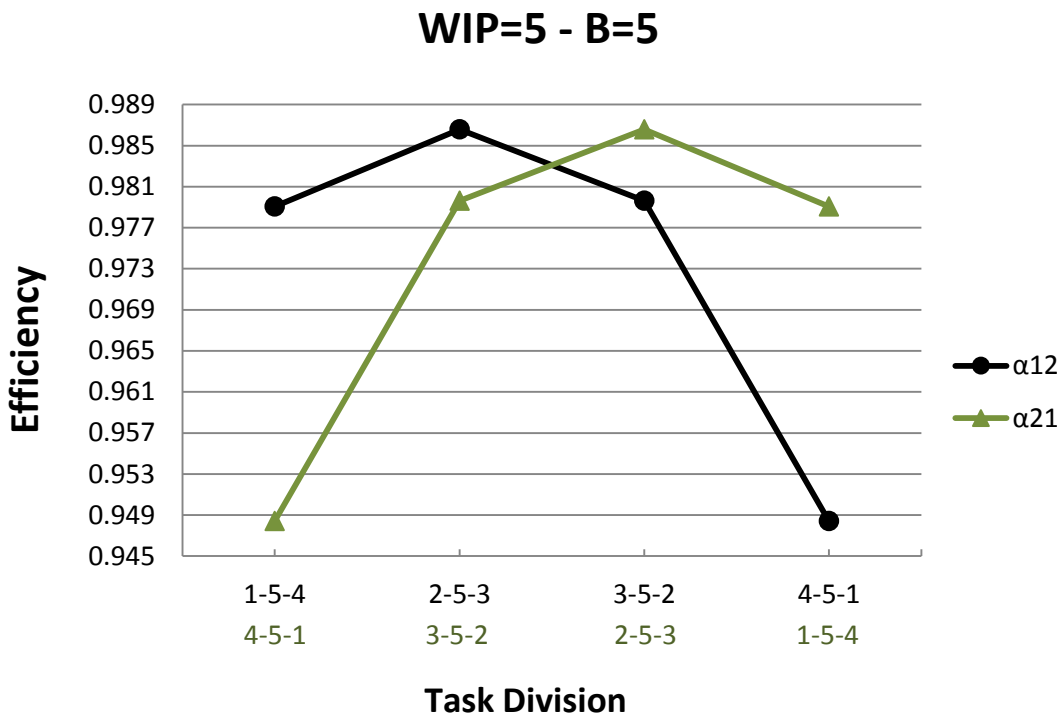


Figure 3-10 *Efficiency with different task divisions and B=5*

We noticed some variances between the examples and the average patterns. The reason is that the efficiency is also affected by the size of the shared task. Figure 3-11 plots the efficiency with different shared task' size which is represented by the fraction of shared task to total work (β) and under (α_{12}=1, α_{21}=0). The performance deteriorates as β increases after 0.2. In this case, station 1 has a big fixed task and as shared task' size gets bigger, additional tasks will be available to W_1. As a result, station 2 starves for longer time. On the other side, there is no remarkable change with (α_{12}=0, α_{21}=1) as the size of shared task changes except for low *WIP* where the performance declines (Figure 3-12). In the low WIP, station 1 is more prone to starvation as β raises than the other cases where the efficiency remains almost the same. In these cases, the station 2 will be the bottleneck, and since W_1 who will decides to pass or keep the shared task, he can adapt with the bottleneck's effect by keeping working on the shared task more frequently and the ample *WIP* can support.

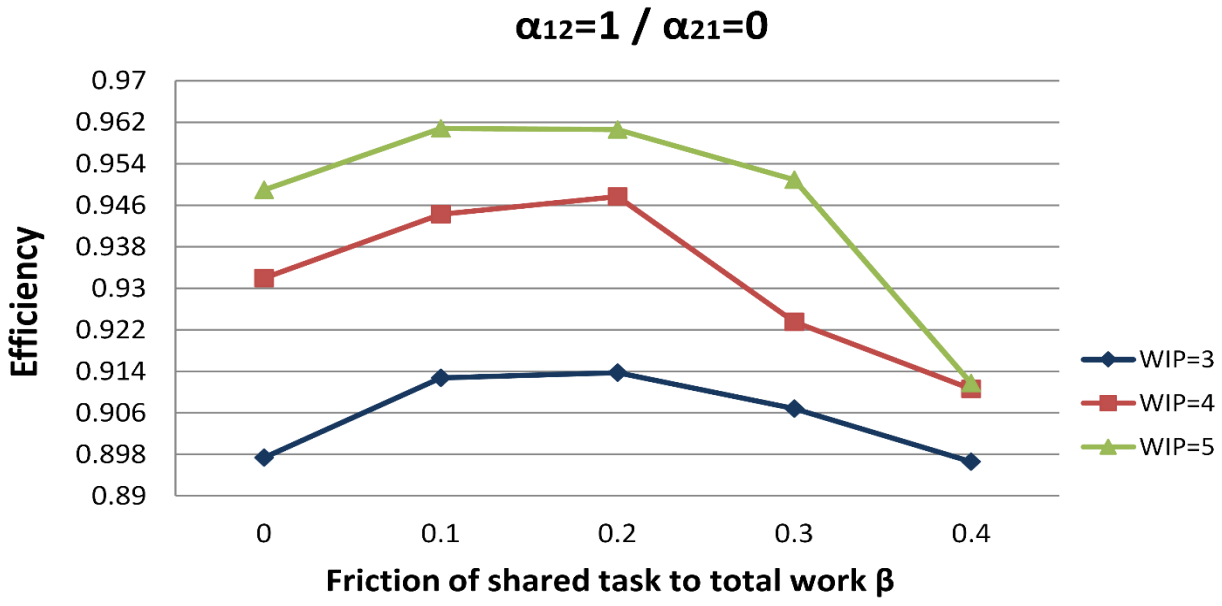


Figure 3-11 *Efficiency versus β under α_{12}=1, α_{21}=0*

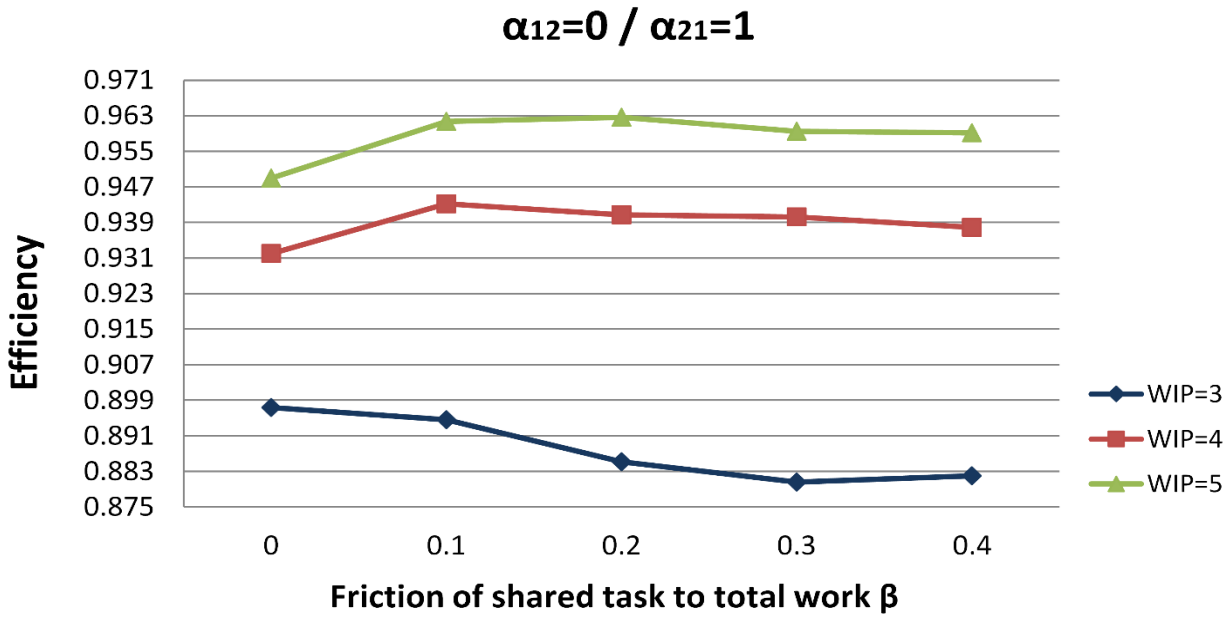


Figure 3-12 *Efficiency versus β under α_{12}=0, α_{21}=1*

The balanced fixed workload (Figure 3-13) which can be translated as $\alpha_{12}=\alpha_{21}=0.5$ has the same trend as in previous papers [3], [4], [5]. The performance improves as β increases till 0.4 then the efficiency declines.

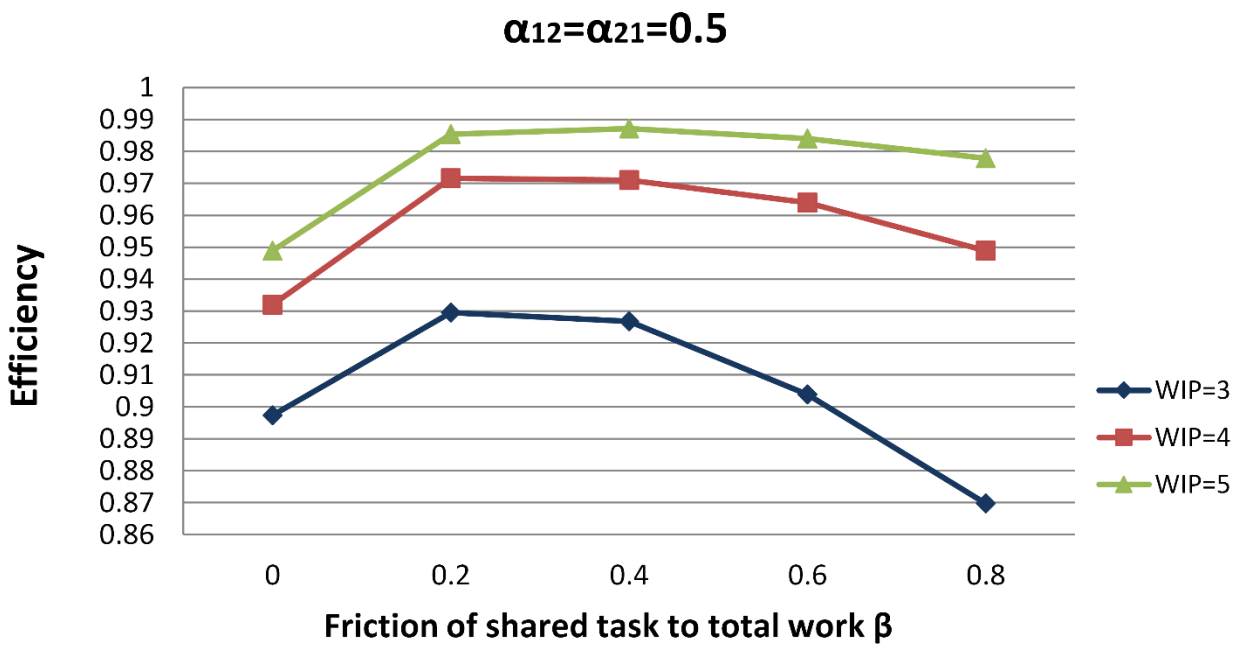


Figure 3-13 *Efficiency versus β under* $\alpha_{12}=\alpha_{21}=0.5$

To summarize the comparison between the opposite configurations, we use the relative difference. For example, D (0-1, 1-0) equals the efficiency of ($\alpha_{12}=0$, $\alpha_{21}=1$) minus ($\alpha_{12}=1$, $\alpha_{21}=0$) then the result is divided by the efficiency of ($\alpha_{12}=0$, $\alpha_{21}=1$).

Figure 3-14 shows the results of these differences. As much as the fixed workload approaches toward the balanced workload, the differences diminishes. A line with *WIP* =3 and 5 has the biggest difference for all cases while the one with a moderate amount of *WIP* has a small difference.

The variance is positive for the big *WIP* where a small value of α_{12} or a small fixed task at station 1 shows better performance. On contrary, a big fixed task at station 1 gives higher efficiency with small *WIP* as we found previously. With moderate *WIP*, the small fixed task at station 1 (α_{12} is close to zero) exhibits better performance for extreme cases while the big fixed

task at station 1 at some certain amount displays higher efficiency for moderate cases of configurations. In extreme cases, the effect of having more *WIP* is more influential where it can offset existing a small fixed task at station 1 and in this case, both stations suffer less starvation.

On the other side, a big fixed task at station 1 at or after the balanced workload point to some extent leads to higher results then drops dramatically when this task gets so big as it hinders flowing the jobs to the station 2. The contribution of two factors, having somehow big fixed task at station 1 and the surplus of *WIP*, helps to improve the performance.

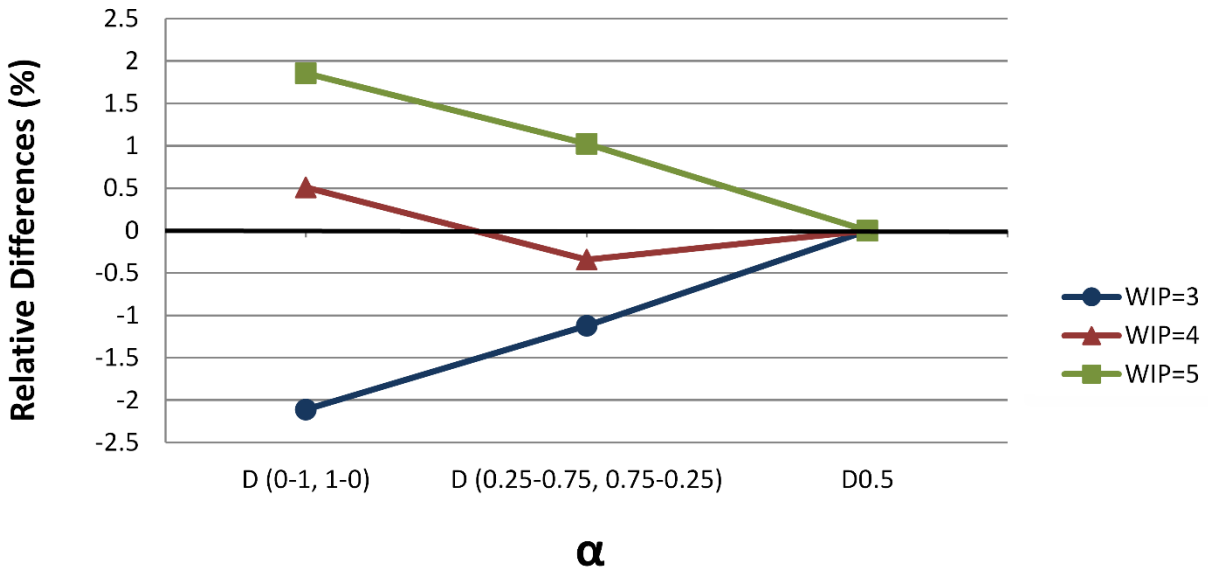


Figure 3-14 *The discrepancy between the opposite configurations*

In the above analysis, we presented the mean values of studied groups. To have more deep view, it is important to show the variation in each group and compare between other groups and different *WIP*. Standard deviation is employed for that purpose. Figure 3-15 indicates three forms of variations according to the amount the jobs in the line. However, in all of these three

the extreme configurations are close to each other for all values of *WIP*. Small *WIP* generally have more variability than others do especially when the fixed workload is far from the extreme cases. The balanced fixed task has the highest variability where the more effective factor is the size of shared task with absence of supportive WIP. When there is more jobs in the line (WIP = 4, 5), the variability decreases dramatically. A line with moderate or large WIP is more stable where all groups show a small value of standard deviation except when the fixed task at station 1 is too big.

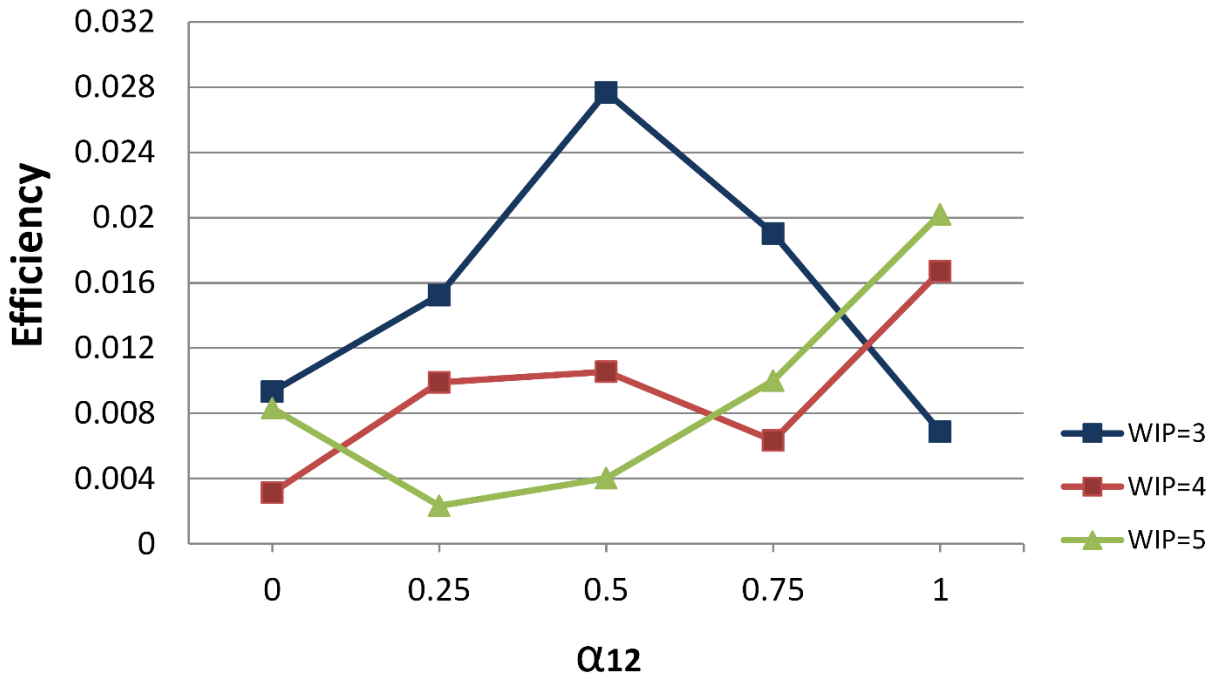


Figure 3-15 *The variation in each studied group*

With large WIP, the variability decreases as the fixed task at station 1 grows until reaching the balanced fixed workload then the variability rises up. The group of configuration with a large fixed task at station 1 has the highest variability in this case.

Considering the variability of the whole job tasks, the workload pattern will suffer a big difference. Conducting experiments with coefficient of variability (CV) of 0.5 instead of 1 as with the above analysis, the outcomes show a notable change when *WIP* is small. Figure 3-16 illustrates a comparison between the performance of two CV (1, 0.5).

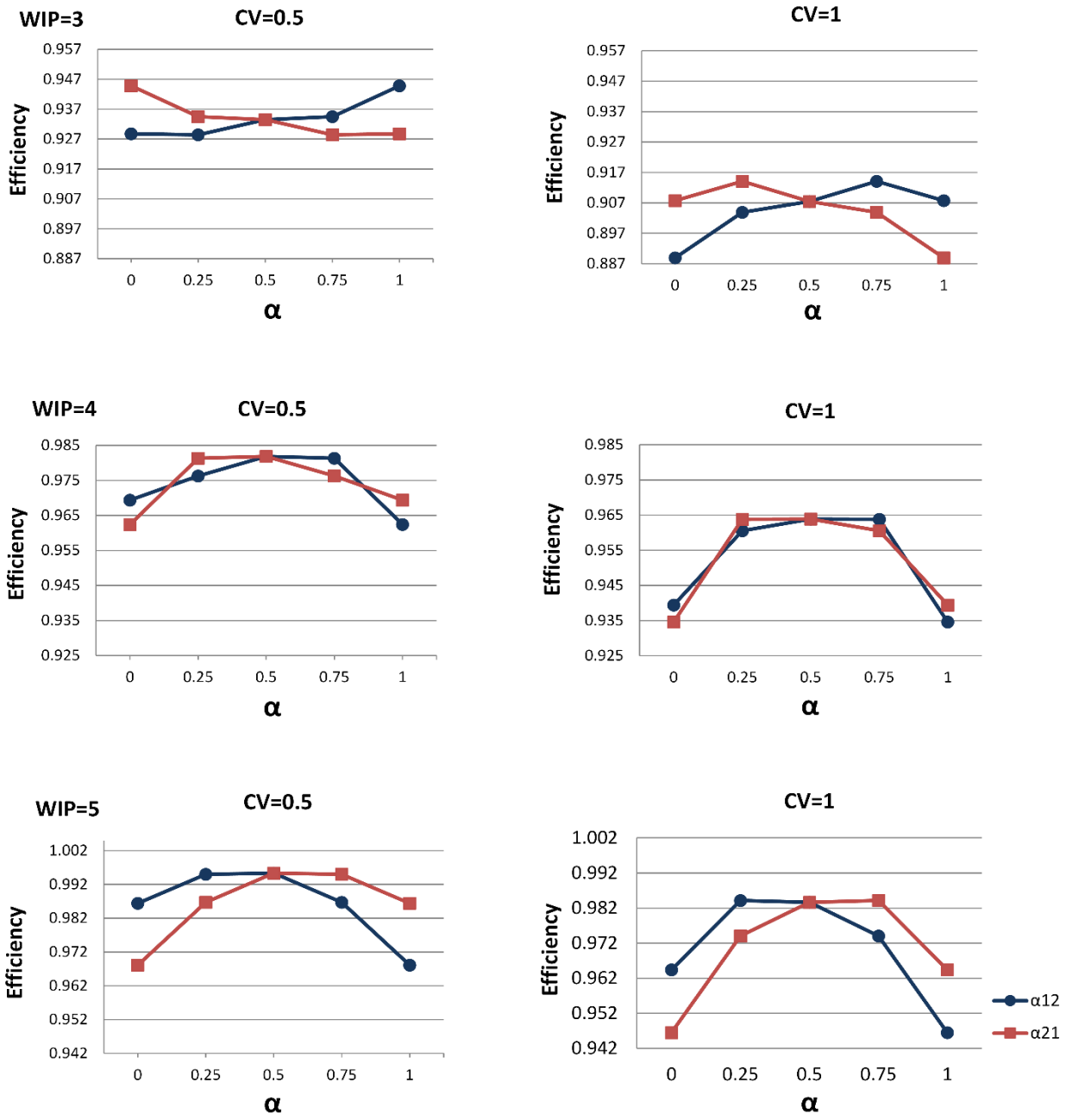


Figure 3-16 *The performance of two levels of variability*

It is noticed, the performance of less variable case is better and the variances between the opposite configurations and the highest and lowest efficiencies are smaller than the case with high CV. As mentioned above, the case with scarce *WIP* experiences a big change in term of the workload pattern. The efficiency of (α_{12}=1, α_{21}=0) gets better when CV is low comparing with high CV. As the number of jobs in the line is small, having a big fixed task at station 1 will make the choice of sending the job with undone shared task to stations 2 more frequent. Beside to that, the low variable process will reduce the potential of W_1 being idle due to the lack of jobs. With high variability, this potential increases.

3.8 Conclusion

We investigated in this research a wide range of workload configurations for a two-station production line. The results of simulation show a remarkable trend of efficiency through different workload configurations. We find that a big fixed task at upstream station with small amount of *WIP* can give better performance than having a big fixed task at the downstream station. As much as *WIP* increases, the upstream bottleneck affects badly the efficiency. The variability has a little to do with the performance pattern while it improves the performance in term of its value and the differences between the different configurations.

A downstream bottleneck with this surplus of WIP supports the throughput comparing with the upstream bottleneck. However, as fixed workload goes toward the balanced status, the results gets close to the maximum. Two factors can be considered in this stream, which can help to improve the performance. The first is the ability to process the job in flexible order, which facilitates the mission of production stuff by altering the position of workers in the line. Another factor is the possibility to divide a task to have a near balanced fixed workload.

References

[1] Ostolaza, J., McClain, J. and Thomas, J., 1990. The use of dynamic (state-dependent) assembly-line balancing to improve throughput. *Journal of Manufacturing and Operation Managemen*t, 3:105-133.

[2] Bartholdi, J.J. and Eisenstein, D.D., 1996. A production line that balances itself. *Operations Research*, 44 (1): 21–34.

[3] Bokhorst, J. A. C., 2011. The impact of the Amount of Work in Process on the Use of Cross-training, *International Journal of Production Research*, 49(11): 3171-3190.

[4] Gel, E. S., Hopp, W. J. and Van Oyen, M. P., 2002. Factors affecting opportunity of worksharing as a dynamic line balancing Mechanism. *IIE Transactions*, 34(10): 847-863.

[5] Askin, R.G., and Chen, J., 2006. Dynamic Task Assignment for Throughput Maximization with Work-sharing. *European Journal of Operational Research*, 168: 853– 869.

[6] Chen, J., and Askin, R.G., 2006. Throughput Maximization in Serial Production Lines with Work-sharing. *International Journal of Production Economics*, 99:88–101.

[7] Spearman, M. and Zazanis, M., 1988. Push and pull productions systems: issues and comparisons. *Department of EE and MS, Northwestern University*, Tech. Rep. 88-24.

[8] Spearman, M.L., Woodruff, D.L. and Hopp, W.J., 1990. CONWIP: a pull alternative to Kanban. *International Journal of Production Research*, 28: 879-894.

[9] Hopp, W. J., and Spearman, M. L., 2008. Factory physics: Foundations of manufacturing management. Third Edtion. *New York: IRWIN/McGraw-Hill*, 365.

[10] McClain, J.O., Thomas, J. and Sox, C., 1992. On-the-fly line balancing with very little WIP. *International Journal of Production Economics*, 27: 283-289.

[11] Alan, A., Pritsker, B., and O'Reilly, J. J., 1999. Simulation with visual SLAM and AweSim. 2nd Edition, *John Wiley & Sons*, New York.

CHAPTER 4

FACTORS AFFECTING THE DLB PERFORMANCE

Overview

This chapter addresses the structure factors that affect the DLB performance with considering the workload imbalance. Three factors are tackled; Information Accuracy, Granularity of shared task and Variability. The results show the change happens in term of the value and the pattern of the performance through the variety of workload configurations.

4.1 Introduction

Generally, the production lines are always prone to many disturbances. That would lead in the most cases to deteriorating the performance. Some factors that cause these deviations far from the target are internal (we focus on the internal factors) and others are external. The internal factors can be overcome by manipulating the characteristics of the elements composing the line. As these factors are internal, they are easier to manage or remove than the external ones. The external factors are out of control in many cases so adapting or at most alleviating their effects is the most the production managers can do.

Exploring these factors and how they influence the performance is essential to tackle them. The treatment might contain eliminating them or at least reducing the severity of their effects.

In the conventional serial production line, the factors that have an impact on the outcome are well investigated in the literature. In DLB as a special way to apply, the work-sharing, new related factors must arise. The affecting factors in this context are rarely or briefly studied. The only paper that considers some of these factors deeper than others is [1]. On the other hand, they focused on a special case of line which is a balanced case. Askin and Chen [2], [3] conducted experiments on the performance of DLB with considering the interaction of two factors; granularity and information accuracy. Their model is also a balanced fixed workload.

We try in this research to explore several factors and investigate their effects under more general model with balance and unbalance fixed workload configurations. These factors are Information Accuracy, Granularity and Variability.

4.2 Information Accuracy

Generally, to manage a production line it is needed several types of information. In work sharing environment, the types of information depends on the work sharing mechanism. For example, a line with full ability of workers to do any task and to move between stations needs different kind or scope of information than a line having non movable workers with some level of work sharing.

Dynamic Line Balancing is used as a mechanism to manage the work-sharing here. In DLB, W_1 after finishing task A has to make a decision whether to pass on or continue processing the shared task B. This decision considers the current state of system. Appropriate decisions [2],[3] will give more work to station 2 when it is more available or potentially so, or pass less work onto it when otherwise. In other words, it tries to reduce the starvation with fewer buffers.

On other side, it is not easy to make the best decision for all cases since this process is state-dependent and complicated to compute and implement in reality [1], [4]. Threshold rules are more practical, where the decision is made based on the comparison between the available downstream workload *Z* and a specific threshold called *R*. Many threshold rules are found in the literature. These rules can be distinguished based on the information accuracy level required. Mostly, one of two levels of the information accuracy is needed to make the decision to pass or keep the shared task. The first level (Low) requires the information about the amount of work available in the downstream buffer, which is easy to know by simply recognizing the number of jobs with and without a completed shared task in the buffer. The more difficult level (High) to identify is specifying the uncompleted work in the buffer and at the next station. To do that, it might be required more sophisticated methods depending on the nature of tasks. These information accuracy levels are employed in threshold rules as described below to facilitate the decision process. Figure 4-1 plots the two levels of information accuracy.

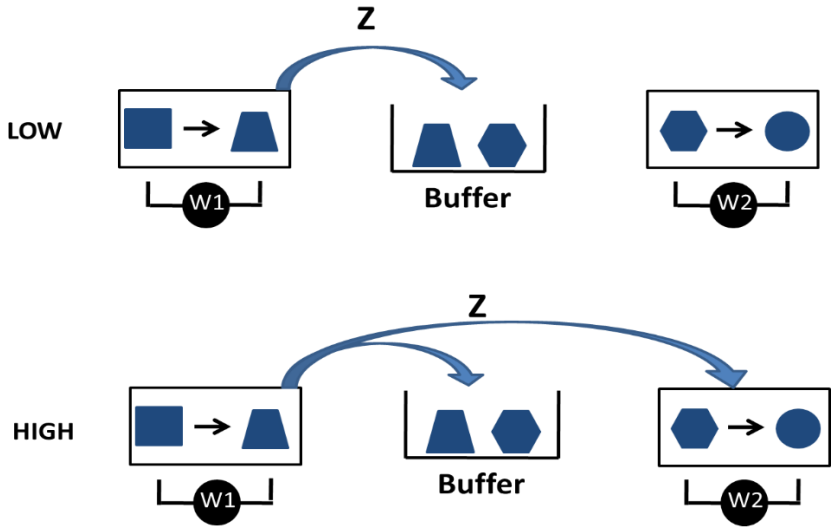


Figure 4-1 *Two Levels of Information Accuracy*

We used one rule from each class that is the most near optimum based on [2]. Of the first level, SRNS is used and of the second HFB is used. In SRNS (HFB) W_1 will start or keep working on the shared task if R or more units of subtasks are available in the buffer before (and at) station 2. W_1 will pass the shared task if the subtasks in buffer before (and at) station 2 are less than R. The threshold value R is given by (1) for SRNS and (2) for HFB, where WIP is the CONWIP level and t_c (t_b) is the number of subtasks for task C (B):

$$R_{SRNS} = \begin{cases} (WIP-2)t_C, & if\ t_B \geq (WIP-2)t_C, \\ [t_B, (WIP-2)t_C] & if\ t_B < (WIP-2)t_C \end{cases} \quad (1)$$

$$R_{HFB} = \frac{t_B}{2} + \frac{(t_B + t_C) + t_C}{2} \cdot \frac{WIP-2}{2} \quad (2)$$

4.2.1 Settings of Experiments

The same model as in the chapter 3 but this time we did the experiments under the two rules SRNS and HFB rules. However, here we prolonged the time of simulation to 900×10^3 time units and increased the number of replications to 12.

4.2.2 Results and Discussion

Figure 4-2 presents the results under HFB and SRNS for WIP =3. We notice that the case with balanced workload (α_{12}=α_{21}=0.5) has better performance in HFB than that in SRNS. The cause returns to that the workload's effect is minimal in this case since each worker has the same probability to starve. As a result, the accurate information used in HFB helps reducing the

opportunity of both workers' starvation by sending the shared tasks as exact as W_2 does need but without causing long starvation for W_1. For example, W_2 has just started processing a job and the buffer before it becomes empty. If at the same time W_1 has finished his fixed task, (in HFB) he will send or not the job with considering how many subtasks still under processing at station 2. If they are enough, the job will not be sent even if the buffer is empty. On the other hand, in SRNS W_1 will surely sends it if the buffer is empty which might make him starving longer.

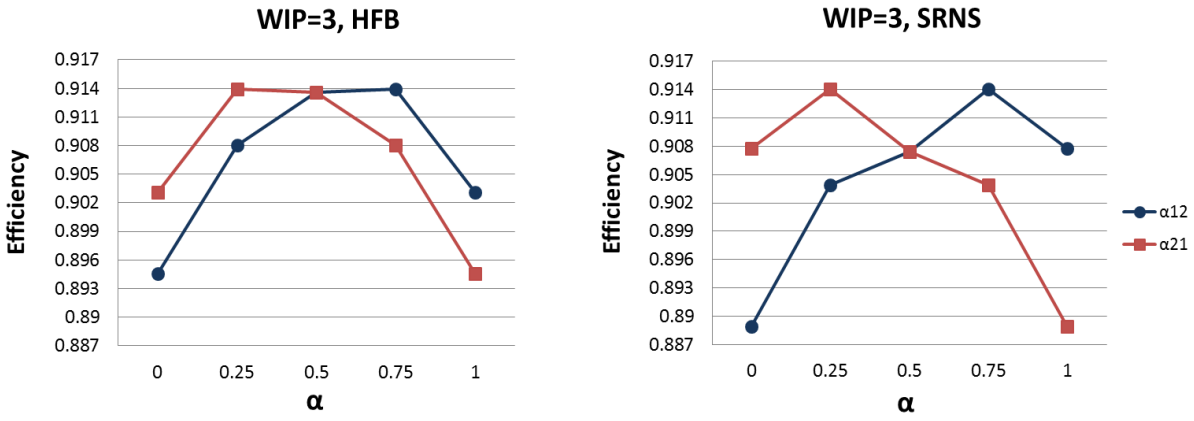


Figure 4-2 *Efficiency vs. α for WIP =3*

Another; interesting point, the performance of (α_{12}=1, α_{21}=0) in HFB is less than the one in SRNS. Also by comparing this combination with (α_{12}=0.25, α_{21}=0.75) in both cases, it is obvious that the results of that combination in HFB is less than (α_{12}=0.25, α_{21}=0.75) . The combination (α_{12}=1, α_{21}=0) has a big portion of subtasks for W_1 (5 subtasks) while relatively small number of subtasks for W_2 and R is small under WIP=3. This means that R could be close to fixed task of W_2. In the light of what mentioned above and since in HFB the uncompleted subtasks at station 2 are considered to make a decision to send or keep. The chance of having

empty in-between buffer is more than in SRNS. As a result, W_2 has a high possibility to starve longer in HFB.

The second case is when *WIP* is 4. Here as in figure 4-3, the differences between the opposite combinations get so small. (α_{12}=0.25, α_{21}=0.75), (α_{12}=0.75, α_{21}=0.25) and (α_{12}=0, α_{21}=1), (α_{12}=1, α_{21}=0) have a small difference in their efficiency.

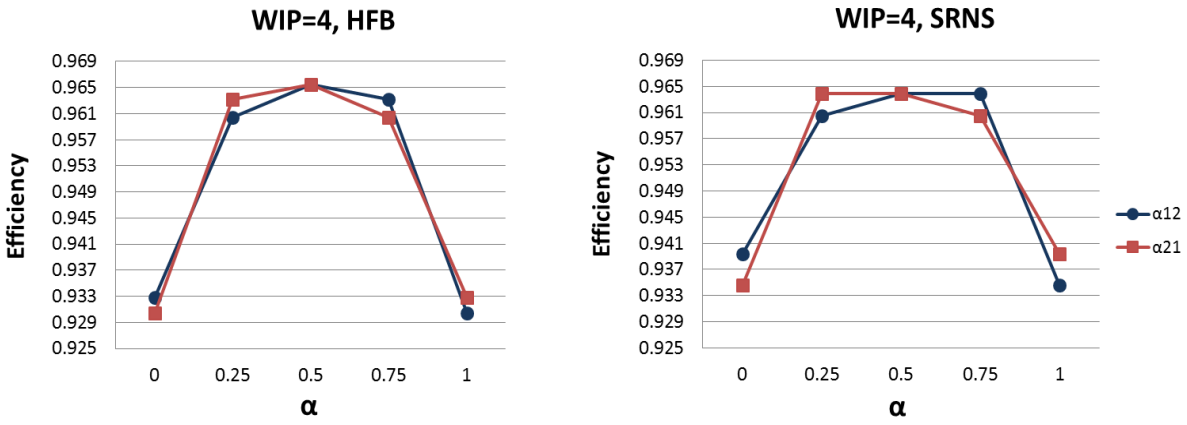


Figure 4-3 *Efficiency vs. α for WIP =4*

For α_{12}=α_{21}=0.5, the performance becomes as same as (α_{12}=0.75, α_{21}=0.25) in SRNS and gets a bit higher in HFB. The surplus of WIP moderates the starvation of *W1* and W_2, which results to minimize the differences between the opposite combinations.

In addition to the above mentioned, we have almost similar patterns for both rules. However, the workload configurations with extreme values of α show a bit better efficiency in SRNS than that in HFB. With high level of information accuracy as in HFB, considering the number of subtasks at station 2 with ones in buffer helps sending no extra subtasks to downstream buffer whereas in SRNS the buffer will be full more, which can be noticed easily

from a bit longer cycle time in SRNS. With an ample *WIP*, a line with SRNS experiences less starvation for station 2 and no negative effect on station1. The effect of this behavior on the performance diminishes as the workload configuration goes toward balanced case (α_{12}=α_{21}=0.5) where each worker has a relatively equal opportunity to starve.

The last case when *WIP* reaches to five. Each rule shows a different pattern of efficiency. In HFB (Figure 4-4), the change of efficiency has almost the same pattern as in the previous case (*WIP*=4) with higher efficiency. The higher level of information accuracy helps to have more robust pattern against the increase of *WIP*.

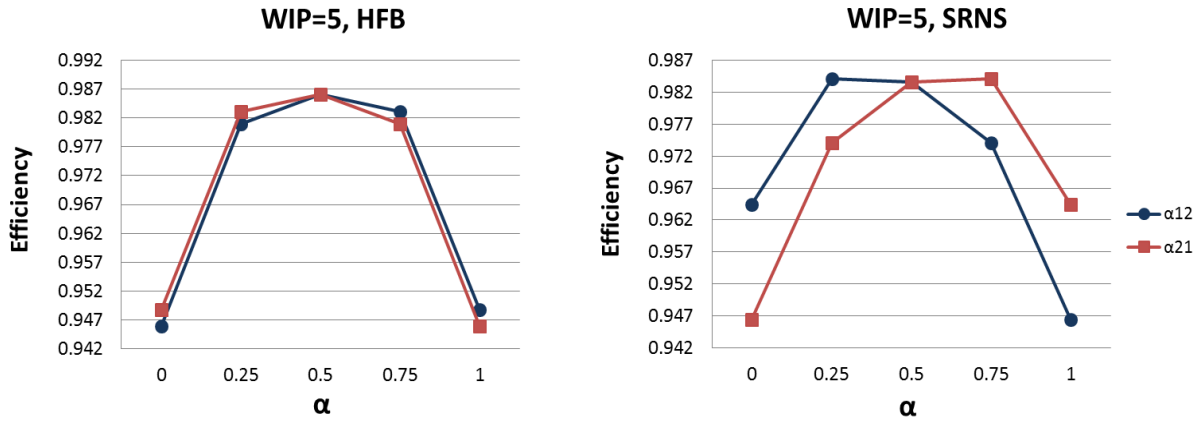


Figure 4-4 *Efficiency vs. α for WIP =5*

On the other side, Figure 3-4 also displays a pattern of SRNS that is opposite to the first case but the differences between the opposite combinations are a bit less.

Another notable point (from figure 4-4) is that the differences between the opposite combinations in SRNS are bigger than that in the previous case and in HFB for the same case. For example, for $\alpha_{12}=1$, $\alpha_{21}=0$ we have a big fixed task at station1 that delays inserting more

jobs and *R* decreases as the fixed task at station 2 shrinks. In this sense, station 2 starves longer thus the efficiency deteriorates. This matter is not existed in HFB and in the case with *WIP*=4 where its effect is tiny since the *WIP* is not so big.

4.3 Granularity of Shared Task

The granularity of shared task can be classified into three types; non-preemptive, preemptive, and between these two extremes. The granularity refers to the number of decision points that divide the shared task into several subtasks. The decision that is made at each of these points is to pass or keep working on the shared subtasks. In this logic, in the preemptive shared task the decision to keep or pass can be made at any point during processing the shared task while in the non-preemptive shared task the decision is made only before processing the shared task starts. In the cases between these two extremes, there are natural break points in the shared task where the shared task can be transferred to the next station only at these points [1]. The majority of research papers depended totally or partly on the non-preempted shared task as in [2-10]. A few papers employed a preempted task for sharing [1], [11-12], whereas the others investigated a range of granularity [1], [2] and [3]. We focus on the last type of studies in our research range.

Gel et al. [1] defined the granularity as "a measure that accounts for the number of break points and the size of subtasks that are transferable in the shared task." They combined these two factors in one metric of granularity (*g*). They used a Markoven Decision Process (MDP) model and got the optimal policy that controls the decision to send or keep the job by using dynamic programming. However, they found that the resulting rule is difficult to apply.

Therefore, they tried to find the optimal heuristic rules for the potential cases. The resulting throughput of the optimal heuristic policy is compared with the one of best static policy and the evaluation measure was opportunity which measures how well the optimal work-sharing policy performs relative to the best static policy. The experiments were done for a two-station line with Constant Work-In-Process (CONWIP) policy and the processing times of subtasks of shared task are identically distributed exponential random variables. Four cases are studied; preemptive shared task, non-preemptive one and two cases with a granularity level of two subtasks; one with two equal subtasks and the other with one short subtask and one long subtask. The results showed that the performance of both policies improves as the shared task becomes more granular and the opportunity increases as the shared task gets less granular. The lowest opportunity is achieved by the shared task with two equal subtasks, and it is better than the case with a preemptive shared task.

Chen and Askin [2], [3] examined different number of break points of shared with equal subtasks for each case. They also examined a two-stage line but for four cases of control policies; HFB, SRNS, and their optimal models which were gotten by the simulation optimization. Three sizes of shared task are investigated; 2, 4, 6 where 2 has two cases; one and two decision points, 4 has three cases one, two and four decision points. For 6, there were four cases; one, two, three, and six decision points. These numbers (2, 4, 6) represent the number of subtasks that composes the shared task since the subtask model [6] is used by the researchers here. The results of simulation confirmed the results of [1]. In addition, they found that larger shared tasks achieve higher efficiency than smaller shared tasks when given higher granularity. This is contrary to the case with non-preemptive shared task where the small to medium sized shared task gives better performance than the large shared task. The results showed also that *R*

values which represents the threshold decreases in the optimal models and the efficiencies of HFB and SRNS rules jump up as the granularity increases but not as in optimal rules.

In the above papers [1], [2], [3], only two factors are considered in the granularity. These two factors are, the number of decision points, and the size of subtasks resulting by existing of these points. In the paper [1], they did their experiments with one, two, infinite numbers of decision points. In addition, equal and non-equal sized subtasks are used. Whereas in [2], [3] more variety of decision points' numbers is taking in account; one, two, three, four, and six. However, the sizes of resulting subtasks are equal.

In this research, we study other new two factors, which are the order of subtasks resulted from decision points and the workload. The order of subtasks here is related to the size of subtasks. In other words, if the shared task has two decision points with two subtasks, one short and one long, the order means which one is chose first to be done or if the sequence is important which subtask should have bigger size the first or the second. Gel et al. [1] examined a case with two subtasks one is 0.25 of the shared task and the other is 0.75. The wonder here is "Is there a difference in the performance whether the first subtask has 0.75 or 0.25 of shared task?" We try to explore this wonder as that will give more insights in efforts of maximizing the efficiency by just managing these subtasks.

4.3.1 The order of Subtasks

Under some level of granularity, the decision points divides the shared task *B* into a number of subtasks, we symbolize these subtasks as B_1, B_2, B_3, B_4... (Figure 4-5). This number can be specified based on how many points there are. For example, if we have one point that

means B is one subtask and the decision to pass or keep is made before starting processing B. If there are three points, the decision is made three times, before the first subtask B_1, before the second B_2 and before the third B_3. The sizes of these subtasks could be equal or not. Our main focus is non-equal ones since the different sizes could raise the query whether the order is important or not.

Equal Shared Subtasks Unequal Shared Subtasks

Figure 4-5 *Granular shared task with equal and unequal shared subtasks*

The order has two meanings based on the ability of doing the subtasks in sequence or not. If the subtasks have to finish in sequence, the order means where the big portion of shared task should be for the first subtask or for the second and so on. Another meaning, when the sequence is not important, is which subtasks should be processed firstly. For instance (Figure 4-6), let B=4 has two subtasks B_1 and B_2. Their sizes are 1, 3 work units. In sequential processing, (B_1, B_2) = (1, 3) is different than (3, 1) while in non-sequential processing B_1 could be 1or 3. In this study, this matter does not affect the results, and it is totally a technical issue influenced by the processing nature of shared task.

Equal Shared Subtasks Unequal Shared Subtasks

Figure 4-6 *Granular shared task with equal and unequal two shared subtasks*

SRNS is utilized as a threshold rule. The level of granularity that is employed in our experiments is with two decision points. In this case, the shared task has two subtasks; B_1, B_2. Different sizes of shared task are utilized. The resulting subtasks are given the all possible combinations of subtasks with all potential sizes under a step of one unit work. All settings of these experiments are presented in table 4-1.

Table 4-1 *The task divisions and the subtask combinations of B used in the study*

A	B	C	SUBTASKS (B1,B2)
3	4	3	(1, 3) - (2, 2) - (3, 1)
2	6	2	(1, 5) - (2, 4) - (3, 3) - (4, 2) - (5, 1)
1	8	1	(1, 7) - (2, 6) - (3, 5) - (4, 4) - (5, 3) - (6, 2) - (7, 1)

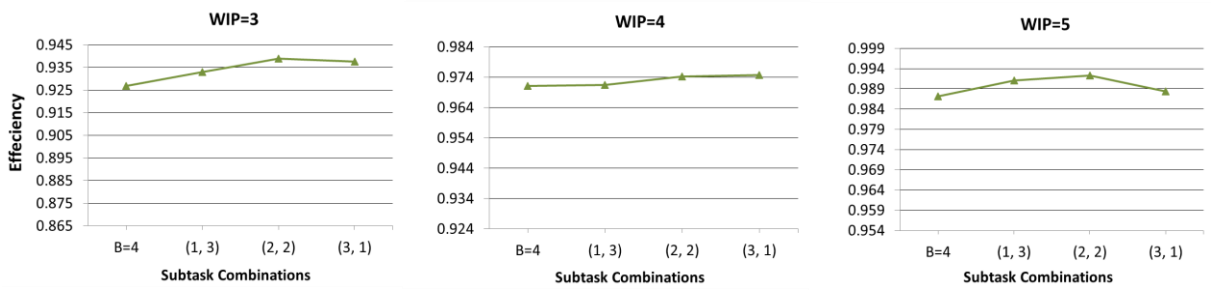


Figure 4-7 *The efficiency with different subtasks' order for task division 3-4-3*

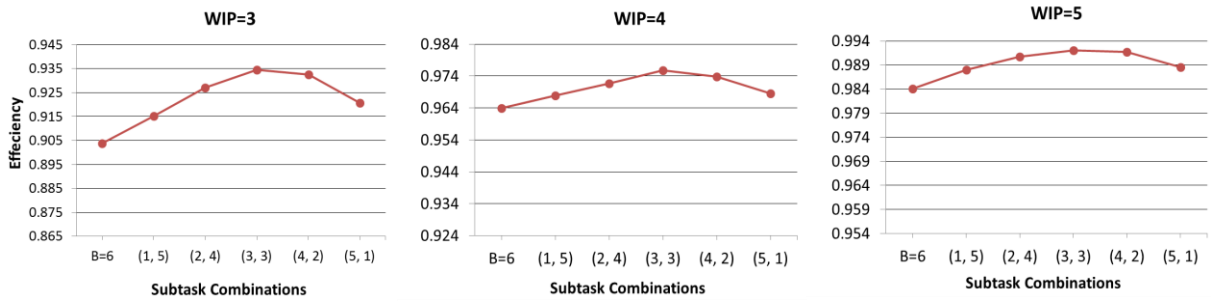


Figure 4-8 *The efficiency with different subtasks' order for task division 2-6-2*

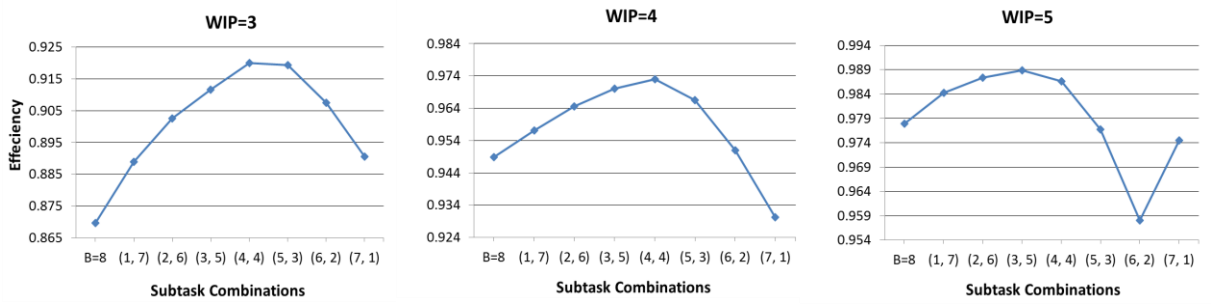


Figure 4-9 *The efficiency with different subtasks' order for task division 1-8-1*

The simulation results of these experiments are plotted in the figures 4-5, 4-6, 4-7. The efficiency is evaluated under three values of *WIP* in the line; 3, 4, 5.

The first observation from the above figures is the efficiency generally improves as the shared task gets granular except for some cases of 1-8-1 under *WIP*=4, 5. This result confirms the outcomes of [1], [2] and [3]. The best efficiency is mostly achieved when the both subtasks have equal size. This is understandable since the fixed tasks for each case have the same number of work units, in this sense the line will be similar to a totally balanced line with benefiting from the DLB advantage in absorbing the effect of variability.

For different order of subtasks, task division 3-4-3 (Figure 4-5) shows no notable increase in the efficiency for all values of *WIP*. All subtasks combinations have almost the same efficiency. The maximum difference between the different order is small around 0.49 % with *WIP*=3 and this difference diminishes as *WIP* gets bigger.

In task division 2-6-2 (Figure 4-6), the variance of efficiency becomes a bit bigger than that in 3-4-3 as the difference between the subtasks' sizes get larger. However, the performance does not show an important change between different orders of subtasks. For example, when *WIP* equals to 3, (1, 5) has less efficiency than (5, 1) by only 0.65%. This tiny variance diminishes as *WIP* rises.

With large shared task as in 1-8-1 (Figure 4-7), the order has more significant effect on the performance. The efficiency of subtasks combinations with an opposite order is almost the same for *WIP*=3 while it has considerable differences for *WIP*=4, 5. The cases with a large first subtask have worse performance than those with small first subtask under *WIP* higher than 3. For example, combinations (2, 6), (6, 2) have efficiency of 0.9645, 0.9509 respectively for

WIP=4. The loss of efficiency is about 1.4%. This can be explained as follows. A line with a large shared task has small *R* which means fewer jobs to send downstream and the potential of having jobs with small shared subtask is higher than the cases without granularity and with big second subtask (when granularity is considered). In this sense, the station 2 will starve more as the first subtask is large even with ample of *WIP* since the station 1 has the decision to keep or send the shared task. In addition, a line with a big shared task has small fixed tasks which promote this starvation.

The difference in the efficiency generally increases as the sizes' variance of subtasks composing the shared task expands. In some cases, the performance even is less than the case with non-granular shared task. For example, for *WIP*=5, the efficiency of non-granular shared task is 0.9779 while of (6, 2) is 0.9580 with loss of 2%. It is clear in Figure 3-7 that the combination (7, 1) has better efficiency than (6, 2), and the difference between (1, 7), (7, 1) is less than that between (2, 6), (6, 2). This can be explained that the combination (7, 1) has the smallest second subtask which makes sending more jobs to approach *R* more potential than in (6, 2). Moreover, the surplus of *WIP* (*WIP*=5) supports this improvement comparing with *WIP*=4 where more jobs are sent downstream as *R* becomes bigger than for *WIP*=4. As a result, the station 2 starves less. In the light of these results, the small to moderate size of shared task does not experience an important effect of different orders of subtasks on the performance. On other hand, in the large shared task the order of subtasks has a significant influence and that becomes clearer as *WIP* increases. The performance in this case (large shared task) gets worse as large as the first subtask becomes.

4.3.2 Workload

We examined a variety of shared task's sizes under a diversity of workload configurations. The sizes of shared task cover three types; small, medium, and large. These sizes are 3, 4, and 6. The workload configurations also include all the potential cases with step of one work unit. To consider the granularity, the experiments are done under the granularity level of two subtasks and different sizes of them. The order of subtasks is not included as it does not notably affect the performance of the studied shared tasks' sizes based on the results of previous analysis. These settings will help to do more rigid analysis and give more comprehensive insights. The workload configurations for each size are displayed in tables 4-2, 4-3 and 4-4.

Table 4-2 *The workload configurations and the subtask combination of B =3*

A	B	C	ST_{12}	ST_{21}	α_{12}	α_{21}	Subtasks
2	3	5	0	3	0	1	(1, 2)
3	3	4	1	2	0.333	0.667	
4	3	3	2	1	0.667	0.333	
5	3	2	3	0	1	0	

Table 4-3 *The workload configurations and the subtask combinations of B =4*

A	B	C	ST_{12}	ST_{21}	α_{12}	α_{21}	Subtasks
1	4	5	0	4	0	1	(1, 3) (2, 2)
2	4	4	1	3	0.25	0.75	
3	4	3	2	2	0.5	0.5	
4	4	2	3	1	0.75	0.25	
5	4	1	4	0	1	0	

Table 4-4 *The workload configurations and the subtask combinations of B =6*

A	B	C	ST_{12}	ST_{21}	α_{12}	α_{21}	Subtasks
1	6	3	2	4	0.333	0.667	(1 ,5) (2, 4) (3, 3)
2	6	2	3	3	0.5	0.5	
3	6	1	4	2	0.667	0.333	

The analysis of the simulation results shows two clear points in the effect of this factor on the performance in the granularity environment. These two are the effect on the granularity's performance, and the influence of granularity on the workload's patterns. We organized the analysis based on the shared task's sizes (small, medium, large) and treated these two points for each size.

In small sized shared task (*B*=3) (Figure 4-8), the extreme values of α_{12}, α_{21}; (0, 1), (1, 0), give the worst efficiency. The performance of α_{12} =1, α_{21}=0 outperforms that of α_{12} =0, α_{21}=1 for *WIP*=3 and the difference between them reaches 4.6%. The reason is that α_{12} =1 means the station1 has a big fixed task (bottleneck) which results having fast processing at station 2 since its fixed task is small. In addition, since the station 1 has the decision to pass or

keep the shared task, it will try to send jobs to the station 2 as many as it is needed. In this way, the station1 will starve less than the case with α_{12} =0 in which the station 2 becomes the bottleneck then fewer jobs are released from the line in any time period. This causes longer starvation for station1 with this small amount of *WIP*.

For *WIP*=4, 5, the previous form gets inversed. The surplus of *WIP* helps to give the advantage to the case with station 2 as a bottleneck. In this case, the station 1 has a small fixed task which allows inserting more jobs in the line than if station1 is a bottleneck. The other values of α have higher performance and the performance of balanced fixed workload is located between these cases. Also, when *WIP* is small, the higher values of α_{12} give better performance.

As *WIP* increases the efficiency of opposite cases gets closer to each other. For example, the difference of efficiency between the extreme cases (α_{12} =1, α_{21}=0), (α_{12} =0, α_{21}=1) reaches 4.6% for *WIP*=3 and then diminishes to 1.5%, and 1% for *WIP*=4, 5 respectively. The performance of extreme values is still the worst while for other cases their performance becomes almost the same for all values. For example, for *WIP*=4, the combinations (α_{12} =0.33, α_{21}=0.667), (α_{12} =0.667, α_{21}=0.33) have the efficiency of 0.977, 0.975 respectively which almost the same.

The pattern of workload does not change comparing with non-granular (NG) shared task. The small size of shared task does not help to have a notable variation, and the efficiency does not have a remarkable increase.

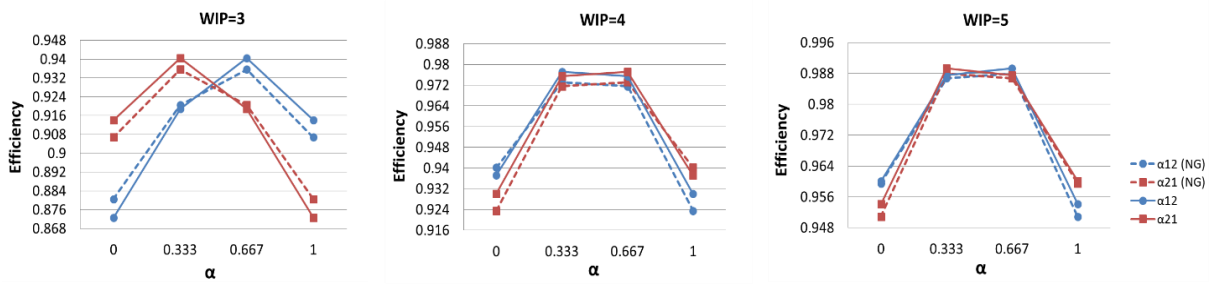


Figure 4-10 *The efficiencies of granularity (1, 2)*

When the shared task increases a bit (medium size *B*=4), the workload configuration has also a distinct influence on the performance in the granularity environment like in the previous case. The pattern of efficiency is the same under different cases of granularity (in our experiments (1, 3) – (2, 2)). Figure 4-9 displays the efficiencies of (2, 2) case. The extreme cases have also the worst performance, and α_{12} =1, α_{21}=0 outperforms α_{12} =0, α_{21}=1 for *WIP*=3 then this gets inversed for more *WIP*. However, the differences between these two combinations have a dissimilar form than the ones in *B*=3 and this variance is large as *WIP* is small and big while it shrinks when *WIP* is moderate. For example, in the granularity of (2, 2) the difference is 4.23% for *WIP*=3 then it reduces to 1.69% for *WIP*=4, and rises again to 4.15% for *WIP*=5.

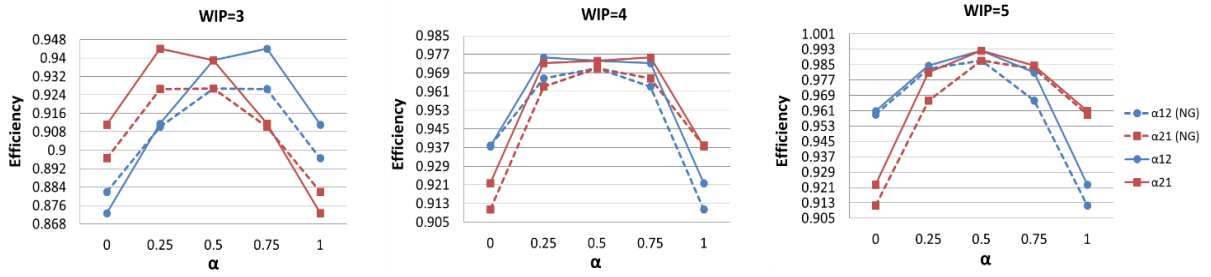


Figure 4-11 *The efficiencies of granularity (2, 2)*

We found that other cases of α have the same style. The performance of balanced workload configuration is located comparing with the two combinations (α_{12} =0.75, α_{21}=0.25), (α_{12} =0.25, α_{21}=0.75) in altered positions as *WIP* changes. It is easily noticed that the performance of balanced workload increases and its position moves from between these two cases to equal to them then higher than them as *WIP* grows. Consider the granularity of (1, 3) as an example. The alteration of balanced workload's performance is presented in table 4-5.

Table 4-5 *The alteration of balanced workload's performance for B=4 and with granularity of (1, 3)*

Combination	Efficiency (E)		
	WIP=3	WIP=4	WIP=5
α_{12} =0.25, α_{21}=0.75	0.9089	0.9714	0.9833
α_{12} =0. 5, α_{21}=0. 5	0.9330	0.9714	0.9911
α_{12} =0.75, α_{21}=0.25	0.9389	0.9710	0.9750

There are some variations between granular and NG shared task in the pattern of workload and these variations are the same for different granularity cases (Figure 3-9). For *WIP*=3, the efficiency of balanced workload is lower than (α_{12} =0.75, α_{21}=0.25) in granular task, while it is equal to this combination in non-granular shared task. For *WIP*=4, the performance of (α_{12} =0.75, α_{21}=0.25), (α_{12} =0.25, α_{21}=0.75) improves comparing with NG task to be equal to the balanced workload. For *WIP*=5, the pattern is almost the same but the difference between the opposite combinations is smaller in granular than NG shared task.

Considering a large sized shared task (*B*=6), the simulation results showed that all different cases of subtasks have the same pattern. Figure 4-10 plots the efficiencies for the

granularity of (3, 3). The figure indicates that the combinations (α_{12} =0.667, α_{21}=0.33), (α_{12} =0.5, α_{21}=0.5) have almost the same best efficiency except for *WIP*=5. If there is a difference between these combinations, it is so small. For example, the maximum difference is 0.43% for *WIP*=3 and the granularity of (2, 2). When *WIP* rises to 5, the combination (α_{12} =0.33, α_{21}=0.667) becomes equal to (α_{12} =0.5, α_{21}=0.5) and they have the best efficiency. In this case, (α_{12} =0.667, α_{21}=0.33) has the worst performance instead of (α_{12} =0.33, α_{21}=0.667). This can be explained as mentioned in the case of *B*=3 and since the fixed workload is small this trend needs more *WIP* to happen.

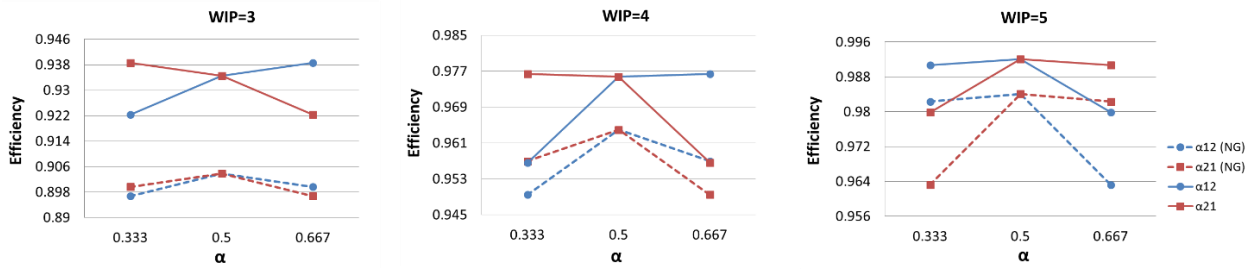


Figure 4-12 *The efficiencies of granularity (3, 3)*

Another obvious point is that the variance of efficiency between (α_{12} =0.33, α_{21}=0.667), (α_{12} =0.667, α_{21}=0.33) is moderate for *WIP*=3, 5 and large for *WIP*=4. For the granularity of (1, 5), this variance is 1.41%, 2.03%, 1.34% for *WIP*=3, 4, 5 respectively. Here, when *WIP* is small, having small fixed task for station 2 helps released jobs faster from the line and that prevents the starvation of station 1 for long time and increases the efficiency. Normally, as *WIP* increases this pattern is expected to get inversed. In other words, having small fixed task of station 1 has higher performance due to the surplus of WIP and inserting more jobs. Unfortunately, this does not happen because (in large shared task *B*=6) the value of *R* rises with *WIP* growing to big

values which means sending more jobs to downstream buffer. For example, for 1-6-3 task division, the threshold value *R* rises from 3 work units for *WIP*=3 to 6 work units for *WIP*=4 while it remains 6 work units for *WIP*=5. This issue plus a big fixed task at station 2 causes the station 1 to starve for longer time than the case with big fixed task at station 1. When we have more *WIP* this matter is solved and the configuration (α_{12} =0.33, α_{21}=0.667) becomes the best as there is enough work units for both stations.

Comparing with NG shared task, the granularity changes the pattern of workload for small to moderate *WIP*. (α_{12} =0.667, α_{21}=0.33) with granular shared task has better performance which is almost equal to the best efficiency. For large *WIP*, the shape of trend is the same for both granular and NG shared task but the (α_{12} =0.33, α_{21}=0.667) gets closer to (α_{12} =0.667, α_{21}=0.33) than in NG shared task.

4.4 Variability

The variability we focus on here is called natural variability. It is caused by minor fluctuations in process time due to differences in operators, machine and materials [13]. In the modeling of processing time variability, we can distinguish between two approaches; task approach and worker approach. The task based approach considers the inherent variability of the tasks is the main source of variability [14]. On the other hand, the worker constitutes the dominant source of the variability. Doerr and Arreola-Risa [15] support this approach. Their field experiments showed that even with identical workers with high variable tasks, the workers are the major source of the variability. In our investigation, we try to consider both approaches in our experiments.

This section addresses the variability in both fixed and shared work. We also consider the disparity of load of fixed work and illustrate the interaction between the workload and variability. The results provide insights for directing the efforts of improving the variability. We try to find answers for the questions like "Having less variable shared work, is it enough to get better performance?", "For some workload allocation, which fixed work should be less variable to have higher throughput?", in other way, "How can we reallocate the workload for some variability combination if the sequence is flexible?".

4.4.1 Settings of Experiments

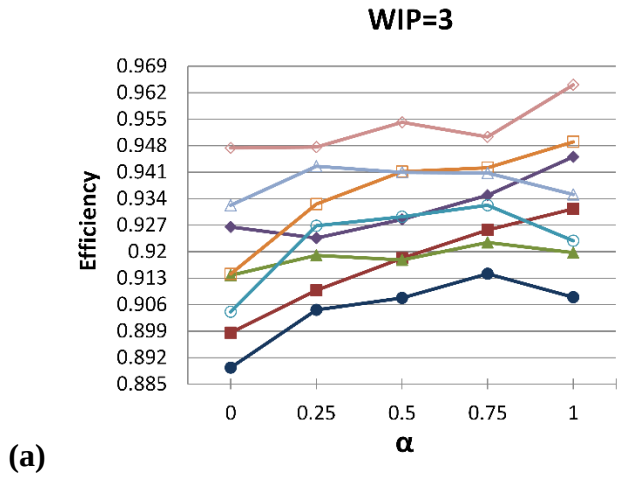
To tackle the variability in this research, we use two levels of variability. These levels are represented by Erlangen distribution with two values of *k* (Erlang-1) for high variability and (Erlang-4) for low variability. The variability combination of tasks is symbolized as V(a,b,c) where a, b and c equal to the value of *k.* For example, V(1,4,4) means that the variability distribution of processing time of a subtask unit is Erlang-1 for subtasks of task A, Erlang-4 for task B and Erlang-4 for task C. Two levels of variability should be examined for each of three distinct tasks so we have eight different combinations of variability. These combinations are displayed in table 4-6.

Table 4-6 *The studied variability combinations*

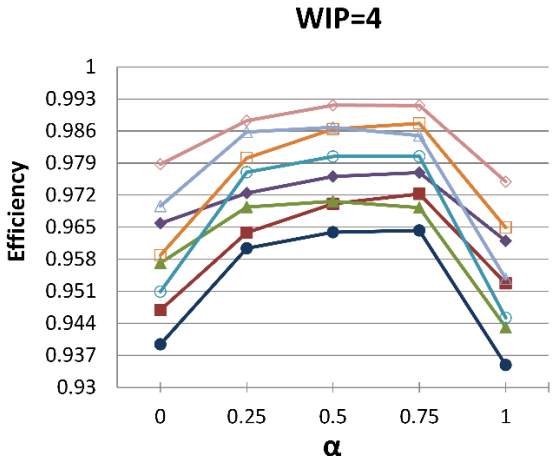
Task\Experiment	1	2	3	4	5	6	7	8
Task A	Erlang-1	Erlang-4	Erlang-1	Erlang-1	Erlang-4	Erlang-4	Erlang-1	Erlang-4
Task B	Erlang-1	Erlang-1	Erlang-4	Erlang-1	Erlang-4	Erlang-1	Erlang-4	Erlang-4
Task C	Erlang-1	Erlang-1	Erlang-1	Erlang-4	Erlang-1	Erlang-4	Erlang-4	Erlang-4
V(a,b,c)	V(1,1,1)	V(4,1,1)	V(1,4,1)	V(1,1,4)	V(4,4,1)	V(4,1,4)	V(1,4,4)	V(4,4,4)

4.4.2 Results and Discussion

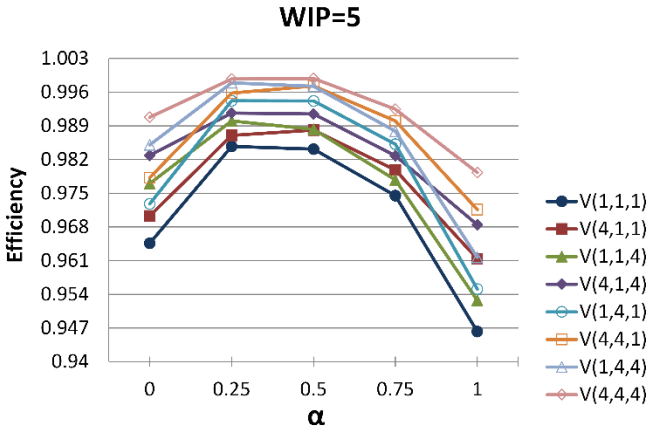
Figure 4-11 and figure 4-12 depict the efficiency of each variability combination as the workload changes in SRNS and HFB respectively. Clearly, the workload pattern when *WIP* is small (*WIP*=3) experience remarkable distortion through different variability combinations. Meanwhile, the pattern is almost the same with tiny change for more *WIP*. Another point, the performance of the same workload configuration improves when the variability of shared task drops except the case of (α =1) (where $\alpha = \alpha_{12}$) where this behavior is a bit different. For instance, V(1,4,4) outperforms V(1,1,4) for any same workload configuration with same variability of fixed tasks. The increase of efficiency between different variability combinations shrinks as the WIP raises since the extra WIP will compensate the variance in variability. The proportions of increase between the highest and lowest performance for balance fixed workload (α=0.5) as an example are 5.12%, 2.88% and 1.49% as WIP raises from 3, 4 till 5.



(a)



(b)



(c)

Figure 4-13 *The performance of each variability combination through workload configurations for SRNS*

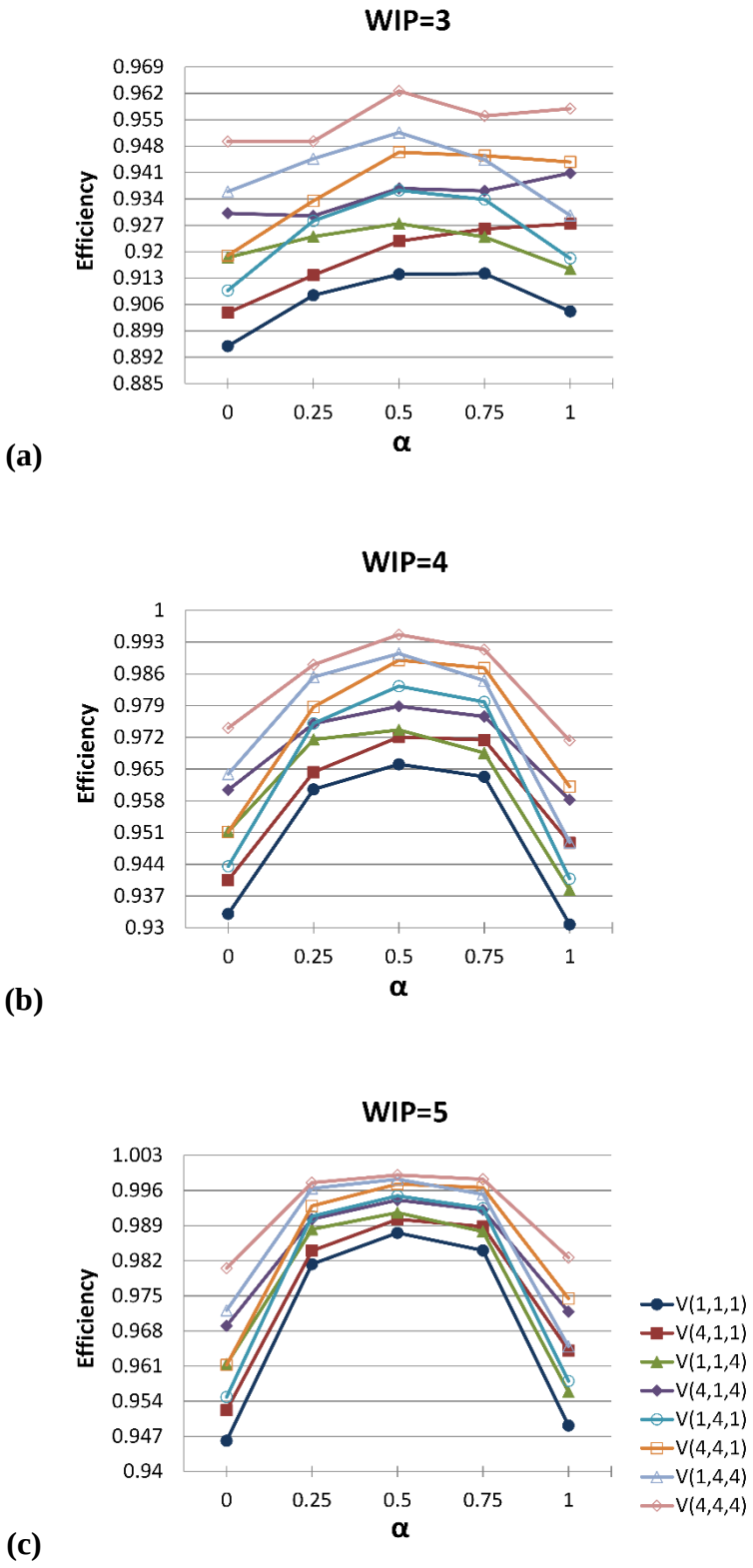


Figure 4-14 *The performance of each variability combination through workload configurations for HFB*

Figures 4-13, 4-14 indicate obviously that there is almost no difference in the performance trend of studied variability combinations between the two used levels of information accuracy and the three WIP cases for each workload alone. It is easy to discover that by checking the style of performance for each workload configuration. The tendency is the same in all charts where the order of variability combinations on the axis ***x*** is same in all graphs. However, the effect of information accuracy is distinct in term of the performance value.

By comparing the efficiency between the workload configurations through all variability combinations, we find three directions. The first one is the balanced configuration (α=0.5). Here, having less variable shared task improves the efficiency dramatically. The line will experience so high performance even better than having task A or task C or both of them less variable. As the second priority in this case, reducing the variability of task C helps to raise the outcome. As CONWIP is closed loop control, decreasing the inter-departure time of jobs leads to shorter inter-arrival time. That minimizes the opportunity of starvation in station 1 especially in low WIP scenarios.

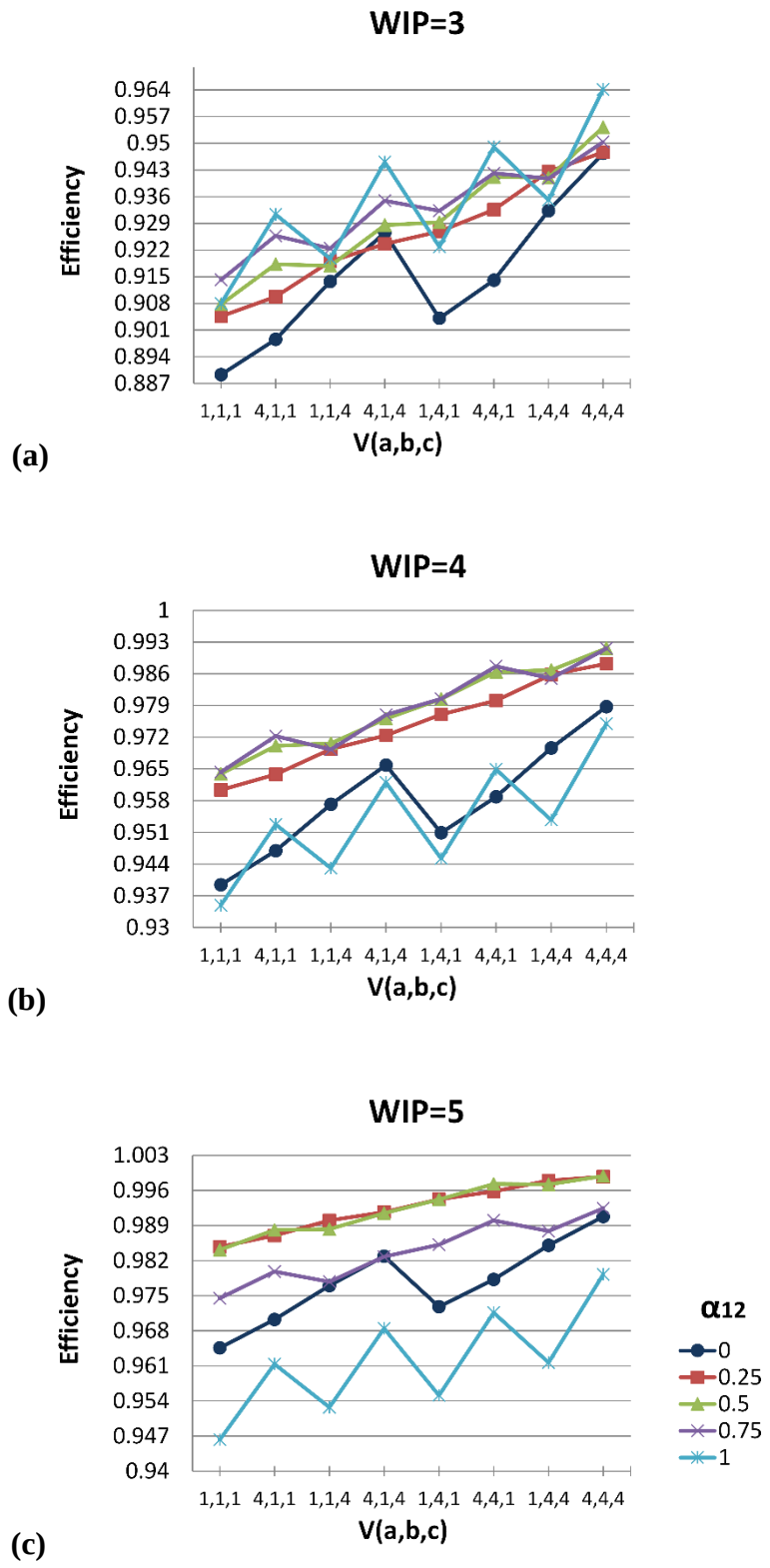


Figure 4-15 *The performance of each workload configuration through variability combination for SRNS*

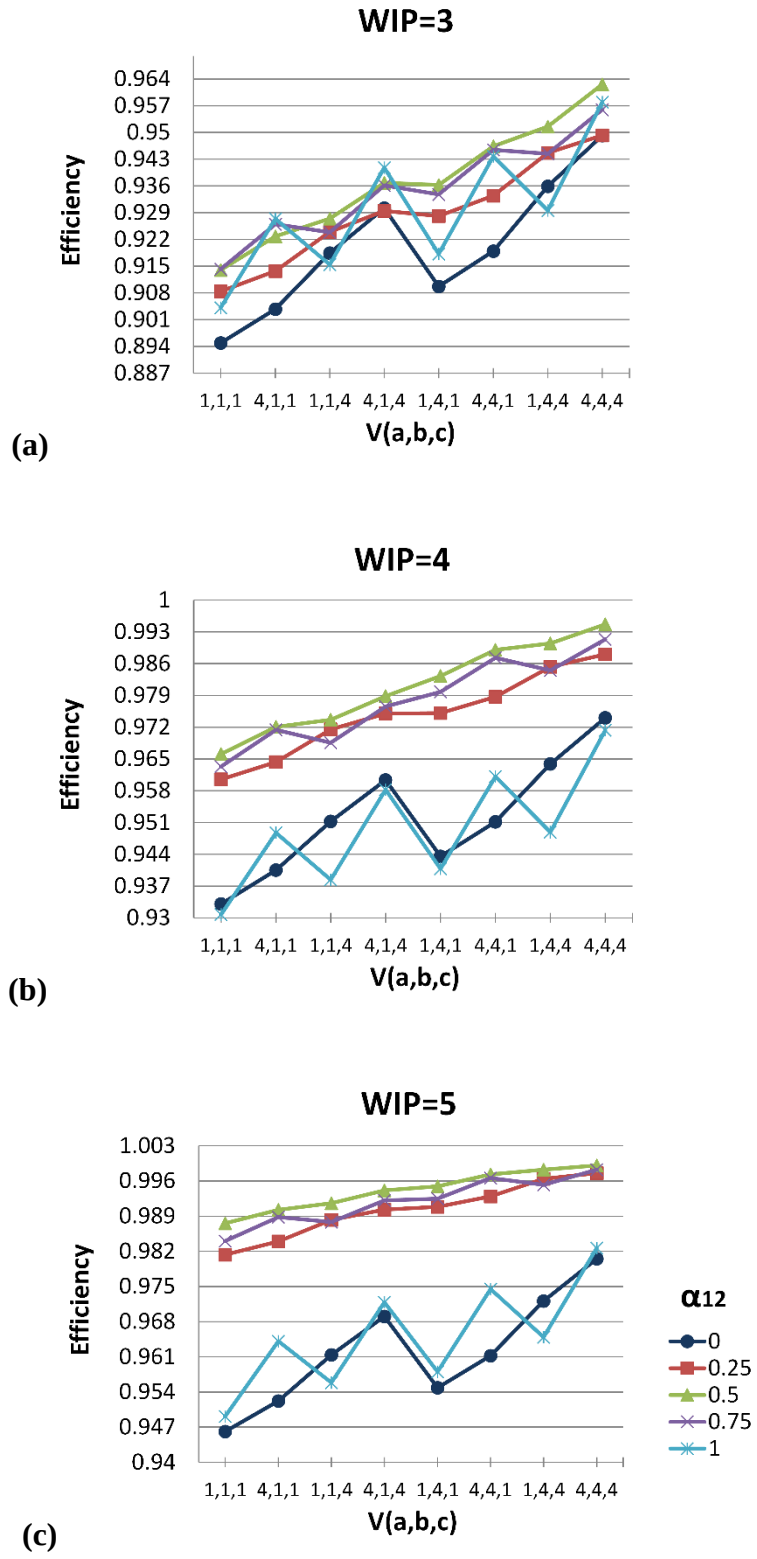


Figure 4-16 *The performance of each workload configuration through variability combination for HFB*

The second and third cases are when the fixed workload of station 1 or 2 goes toward a heavy load. In other words, it becomes a bottleneck. Moderate bottleneck has so similar trend of performance to the balanced case. When the bottleneck has an excessive load, minimizing the variability of the bottleneck is even better than having a low variable shared task. For instance, when α_{12}=1 the efficiency of V(4,1,1) surpasses the one of V(1,4,1). We can also notice that the differences in the performance increase as *WIP* increases for the extreme configurations, which is opposite to the other configurations. The variance is bigger in the case with a high level of information accuracy as in HFB.

4.5 Conclusion

We simulated a two-station production line run under DLB policy. Three factors were investigated information accuracy, granularity and variability. Several workload configurations were employed.

We considered two levels of information accuracy represented by two threshold rules; HFB, SRNS. The simulation outcomes showed that the workload has an obvious effect on the efficiency and this effect has remarkable patterns with several values of α under different values of *WIP*. The information accuracy also affects the performance especially for high and low *WIP* and low information accuracy level generally presents better performance for extreme values of α. On the other hand, higher level of information accuracy shows a better performance as close as the fixed tasks approach to the balance state.

In addition, the high level of information accuracy displays low disparities between the opposite configurations of workload. For instance, (α_{12}=0, α_{21}=1), (α_{12}=1, α_{21}=0) have almost

the same or so close performance in HFB comparing with that in SRNS. In other words, in case of exchanging the position of fixed tasks, high information accuracy shows more robust performance.

Two factors in the granularity environment were explored, the order of subtasks resulting from the granularity and the workload. Both factors have a significant influence. The order of subtasks presents an important effect when the shared task's size is large while this effect is minor for small to moderate size. For the large size of shared task, starting with a small subtask instead of a large one gives a higher performance as WIP increases. The gain could be more than 2.5%.

We also examined the effect of workload that has more vital influence than the previous factor. Different cases of granularity of shared task and workload configurations are studied. The outcomes revealed that there is a pattern of workload configurations for each size of shared task and this pattern has some changes affected by the WIP amount. The granularity affects the workload pattern in term of the value and trend.

Several variability combinations were employed as the last factor in this chapter. We found that generally the shared task should be given the first priority to reduce its variability then the fixed tasks. However, the state of fixed tasks toward the balanced case can affect this general outcome. As much as the fixed workload goes far from the balanced case, focusing on the bottleneck station becomes more important to increase the line performance. At this case, less variable shared task alone gives a small improvement.

References

[1] Gel, E. S., Hopp, W. J. and Van Oyen, M. P., 2002. Factors affecting opportunity of worksharing as a dynamic line balancing Mechanism. *IIE Transactions*, 34(10): 847-863.

[2] Askin, R.G., and Chen, J., 2006. Dynamic Task Assignment for Throughput Maximization with Work-sharing. *European Journal of Operational Research*, 168: 853– 869.

[3] Chen, J., and Askin, R.G., 2006. Throughput Maximization in Serial Production Lines with Work-sharing. *International Journal of Production Economics*, 99:88–101.

[4] Hopp, W. J.,Tekin, E., and Van Oyen, M. P., 2004. Benefits of Skill Chaining in Serial Production Lines with Cross-trained Workers. *Management Science*, 50(1): 83-98.

[5] Ostolaza, J., McClain, J., and Thomas, J., 1990. The Use of Dynamic (state-dependent) Assembly-Line Balancing to Improve Throughput. *Journal of Manufacturing and Operation Management*, 3: 105-133.

[6] McClain, J.O., Thomas, J., and Sox, C., 1992. On-the-fly Line Balancing with Very Little WIP. *International Journal of Production Economics*, 27: 283–289.

[7] Darwin, J. D., Hemant, V. K., and Bret J. W., 2009. Influence of Workload Imbalance on the Need for Worker Flexibility. *Computer and Industrial Engineering*, 57: 319-329.

[8] Gel, E. S., Hopp, W. J., and Van Oyen, M. P., 2007. Hierarchical Cross-training in Work-in-process-constrained Systems. *IIE Transactions*, 39: 125-143.

[9] Bokhorst, J. A. C., 2011. The impact of the Amount of Work in Process on the Use of Cross-training. *International Journal of Production Research*, 49(11): 3171-3190.

[10] Jordan, W.C., Inman, R.R., and Blumenfeld, D. E., 2004. Chained Cross-training of Workers for Robust Performance. *IIE Transactions*, 36: 953-967.

[11] Sennott, L. I., Van Oyen, M. P., and Iravani, S. M. R., 2006. Optimal Dynamic Assignment of a Flexible Worker on An Open production Line with Specialists. *European Journal of Operational Research*, 170: 541-566.

[12] Zavadlav, E., McClain, J. O., and Thomas, L. J., 1996. Self-buffering, Self-balancing, Self-flushing Production Lines. *Management Science*, 42(8): 1151-1164.

[13] Hopp, W. J., and Spearman, M. L., 2008. Factory physics: Foundations of manufacturing management. Third Edtion. *New York: IRWIN/McGraw-Hill*, 365.

[14] Bentefouet, F., and Nembhard, D. A., 2013. Optimal flow-line conditions with work variability. *International Journal of Production Economics*, 141: 675-684.

[15] Doerr, K.H., and Arreola-Risa, A. 2000. A worker-based approach for modeling variability in task completion time. *IIE Transactions*, 32: 625‑636.

CHAPTER 5

MODIFICATION OF COORDINATE RULES WITH CONSIDERING THE WORKLOAD

Overview

After exploring the role of workload on the performance, we suggested a modification in the control rule HFB, which takes in account the workload effect. This chapter aims to elaborate this suggestion and how including the workload in the rule would affect the performance.

5.1 Introduction

In the previous chapters, a thorough investigation has been given covering the workload effect on the line with work sharing applied. Most of the works have dealt with the structural elements comprised the line. For the fixed task, the effect of its size, position, and variability on the performance are studied. On the other hand, different types of aspects have been focused on for the shared task. The granularity of shared task, its variability have been considered. In this chapter, we study the control rule.

Most of rules that use to conduct the work sharing, relay on one idea. This idea is in order to make a decision of keeping or sending the shared task, information about the status of downstream stations is required. The state of downstream might include whether the downstream worker being idle or busy [1], [2], [3] and the amount of available work whether in buffer or in buffer and at downstream stations which is undone yet [4].

Among these rules we target the rules that relay on the threshold value and calculate the cut off value as the half of the available workload for the two neighboring workers. These rules have showed a relatively better performance than the performance in other rules.

Since the main concentration in the DLB literature has been on the balanced case of workload, these kind of rules indicate a distinct performance. Even with imbalance workload, they are able to offset the loss in the yield due to the imbalance workload [4].

5.2 Half workload based rules

In these rules, the considered workload that should be half of is the cutoff or the threshold value is varied. The following rules present variety of the considered workload (t_A the subtask number of task A- t_B the subtask number of task B- t_C the subtask number of task C - *WIP* the number of jobs in the lines):

1- $$R = \frac{B}{2} \quad (1)$$

Where B is the capacity of buffer

This rule is the first version of Half Full Buffer rule. It is introduced by [5] in 1990. The idea here is that the cutoff value is the half of buffer capacity.

2- $$R = \frac{t_B}{2} + \frac{(t_B + t_C) + t_C}{2} \cdot \frac{WIP - 2}{2} \quad (2)$$

This rule is the most recent version of rule (1). It is the near-optimal rule [4], [6]. The second term is the average work content in a half full buffer and the first term represent the amount of work that will equalize the expected idling time of W_1 and W_2 [7].

3- $$R = (WIP - 2) \cdot \frac{t_B + t_C}{2} \quad (3)$$

WIP - 2 is the maximum number of jobs in the buffer if neither station is starved and $t_B + t_C$ is the expected task time for a job with undone shared task. Thus, (*WIP*-2)($t_B + t_C$) defines the maximum workload in the buffer.

4- $$R = (WIP - 2) \cdot \frac{(1-w).t_B + t_C}{2} \quad (4)$$

It takes into account a balance factor that is adapted from Gel et al. [8]. This factor, w, is the solution to the following equation :

5- $$t_A + w.t_B = (1-w).t_B + t_C \quad (5)$$

w estimates the proportion of task B that will be completed at station one over the long run to balance workload.

6- $$R = \frac{WIP.(t_A + t_B + t_C) - t_A}{2} \quad (6)$$

The workload in the system is maximized when all N jobs are present at station one. At the decision point, the maximum workload as seen by station one is WIP. ($t_A + t_B + t_C$) - t_A because one task A has been completed

7- $$R = \frac{t_B + WIP.t_C}{2} \quad (7)$$

The workload in the system is minimized when *WIP* - 2 of jobs with shared task done are at station two. The minimal workload in the system that station one sees at the decision point is t_B + *WIP*. t_C.

8- $$R = \left(\frac{WIP.(t_A + t_B + t_C) - t_A}{2} + \frac{t_B + WIP.t_C}{2} \right) / 2 \quad (8)$$

This rule simply takes the average of maximum-based and minimum-based cutoff levels.

The rule (2) is the best choice to do our study here. Based on [4], the rules from 3 to 7 have the same or worst performance comparing with the rule (2). Accordingly, we made our suggestion to involve the workload in the rule on this rule.

5.3 HFB Rule with involving Workload Variance

Two points motivated us to reform HFB rule by considering the fixed workload imbalance. The first point is that most of papers employing this rule utilize models with balanced workload. The other point is that HFB rule divides the buffer contents into half to get the cutoff value. This point is related to the first point.

Since we did our investigation on the balanced and unbalanced fixed workload, we involved this sense of imbalance into HFB rule. By that, the rule becomes more generalized and considering the imbalance of fixed workload in the rule reflects more accurate information about the line status.

The basic idea is that instead of having the cutoff value by dividing the buffer contents by 2 (half of it), we get this value by dividing the buffer contents by the ratio of fixed workload. In the light of this modification, the shared task should be sent more frequently to the buffer if the fixed task of the downstream station is small and the fixed task of the upstream station is big. Conversely, the shared task should be done more often at the upstream station if the downstream fixed task is big and the upstream one is small. In HFB rule, the buffer contents in equal regardless of the magnitude of fixed workload of up- or down-stream station.

The workload ratio is given as in the following equation (9);

$$w = \frac{t_A + t_C}{t_A} \tag{9}$$

HFB rule after including the workload ratio becomes as follows;

$$R = \frac{t_B}{w} + \frac{(t_B + t_C) + t_C}{w} . \frac{WIP - 2}{2} \tag{10}$$

We call this rule after the modification WFB (Workload-Full-Buffer) since it does not represent the half of the full buffer.

Let us take the following example to illustrate the calculations and the work mechanism mentioned above. Having task division 2-5-3 with *WIP*=4, *w* and *R* are calculated according to equations (9) and (10) as follows

$$w = \frac{2+3}{2} = 2.5$$

$$R = \frac{5}{2.5} + \frac{(5+3)+3}{2.5} . \frac{4-2}{2} = 6.4$$

Taking the opposite case 3-5-2, the *w* and *R* are 1.67, 8.4 respectively. It is easy to notice that *R* is bigger in the case 3-5-2 where t_C is small and t_A is big than 2-5-3 where t_C is big and t_A is small. In 3-5-2, more subtasks should be sent to offset the small amount of downstream fixed task thus *R* is bigger.

5.4 Settings of Experiments

The same model as in the previous chapter is utilized. Three rules are compared; WFB, HFB, SRNS rules. We included SRNS rule since we found in the previous chapter that this rule

showed better performance in some cases than HFB rule. In this sense, this inclusion will present the advantage of the new rule even when HFB failed to give a good performance comparing with a rule under a poor level of information accuracy. The values of *R* for these rules under the three values of *WIP* are present in table 5-1.

Table 5-1 *The cutoff values of studied workload configuration under WFB, HFB and SRNS*

t_A	t_B	t_C	WFB			HFB			SRNS		
			3	**4**	**5**	**3**	**4**	**5**	**3**	**4**	**5**
4	1	5	2.89	5.33	7.78	3.25	6	8.75	1	1	1
3	2	5	3.00	5.25	7.50	4	7	10	2	2	2
2	3	5	2.71	4.57	6.43	4.75	8	11.25	3	3	3
1	4	5	1.83	3.00	4.17	5.5	9	12.5	4	4	4
5	1	4	3.06	5.56	8.06	2.75	5	7.25	1	6	10
4	2	4	3.50	6.00	8.50	3.5	6	8.5	2	2	5
3	3	4	3.64	6.00	8.36	4.25	7	9.75	3	3	5
2	4	4	3.33	5.33	7.33	5	8	11	4	4	5
1	5	4	2.30	3.60	4.90	5.75	9	12.25	4	5	5
5	2	3	3.75	6.25	8.75	3	5	7	2	6	9
4	3	3	4.29	6.86	9.43	3.75	6	8.25	3	4	7
3	4	3	4.50	7.00	9.50	4.5	7	9.5	3	4	4
2	5	3	4.20	6.40	8.60	5.25	8	10.75	3	5	5
1	6	3	3.00	4.50	6.00	6	9	12	3	6	6
5	3	2	4.64	7.14	9.64	3.25	5	6.75	2	3	6
4	4	2	5.33	8.00	10.67	4	6	8	2	4	5
3	5	2	5.70	8.40	11.10	4.75	7	9.25	2	4	5
2	6	2	5.50	8.00	10.50	5.5	8	10.5	2	4	6
1	7	2	4.17	6.00	7.83	6.25	9	11.75	2	4	6
5	4	1	5.83	8.33	10.83	3.5	5	6.5	1	2	3
4	5	1	6.80	9.60	12.40	4.25	6	7.75	1	2	3
3	6	1	7.50	10.50	13.50	5	7	9	1	2	3
2	7	1	7.67	10.67	13.67	5.75	8	10.25	1	2	3
1	8	1	6.50	9.00	11.50	6.5	9	11.5	1	2	3

5.5 Results and Discussion

In general for all studied levels of *WIP (as plotted in figure 5-1),* the new rule shows the superior performance comparing with other two rules with one exception when α=0.75 with *WIP* =4. Nevertheless, the loss of efficiency is so small.

The performance of WFB when α=0.5 is the same as in HFB. WFB turns to HFB when the fixed workload is balanced. The efficiency of correspondent extreme cases have so close value to each other. The maximum difference is 0.36% for *WIP*=3. Meanwhile, the other rules SRNS and HFB are suffering a loss of efficiency about 2.1, 1 % respectively.

For moderate fixed workload imbalance, WFB indicated the superior or same performance. However, this superiority is less than in the extreme cases. As we mentioned in the above paragraph, WFB is more robust against of the position of some task in the line. For instance, the performance of α=0.25 and α=0.75 under *WIP*=5 is the same while SRNS and HFB experience different efficiency according to the different α. SRNS presents a better performance than HFB when α=0.25 while it deteriorates and get inferior to HFB's outcome when α=0.75 as an opposite configuration.

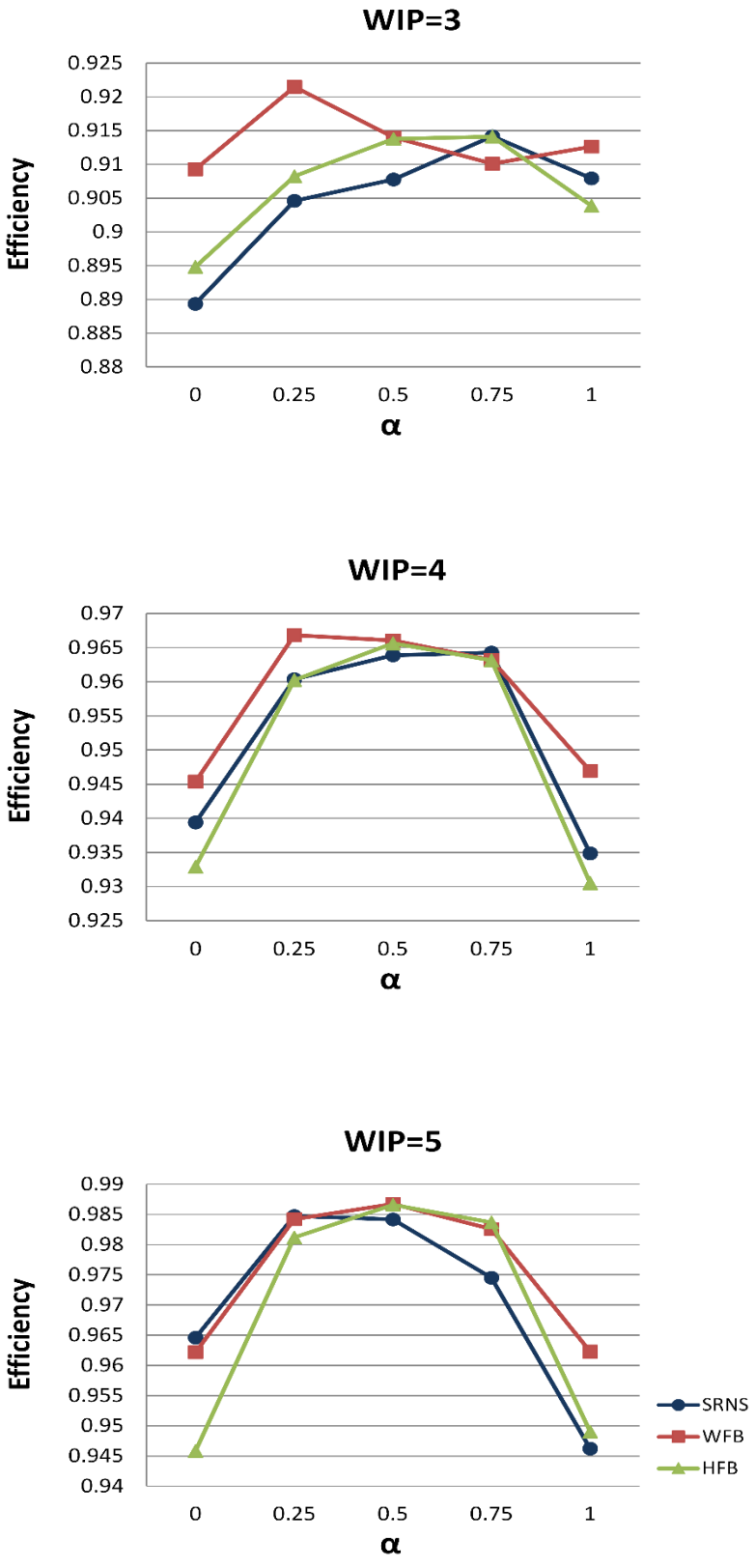


Figure 5-1 *The performance of WFB, HFB and SRNS*

Figure 5-2 plots the gain and loss of efficiency due to applying WFB comparing to HFB and SRNS. As mentioned before, the most of cases are above the bold line indicating to zero improvement rate. The pattern of improvement rate is too affected by asymmetry and sensitivity of others two rules to the task location. When their performance gets better, the rate shrinks.

Taking WFB and SRNS, the contribution of WFB has a big variance as *WIP* changes when the fixed task of W_2 is big. The gain due to WFB increases as *WIP* decreases. Conversely, the line achieves less improvement when W_1 holds big fixed task. Here, the gain due to WFB increases as the number of jobs increases.

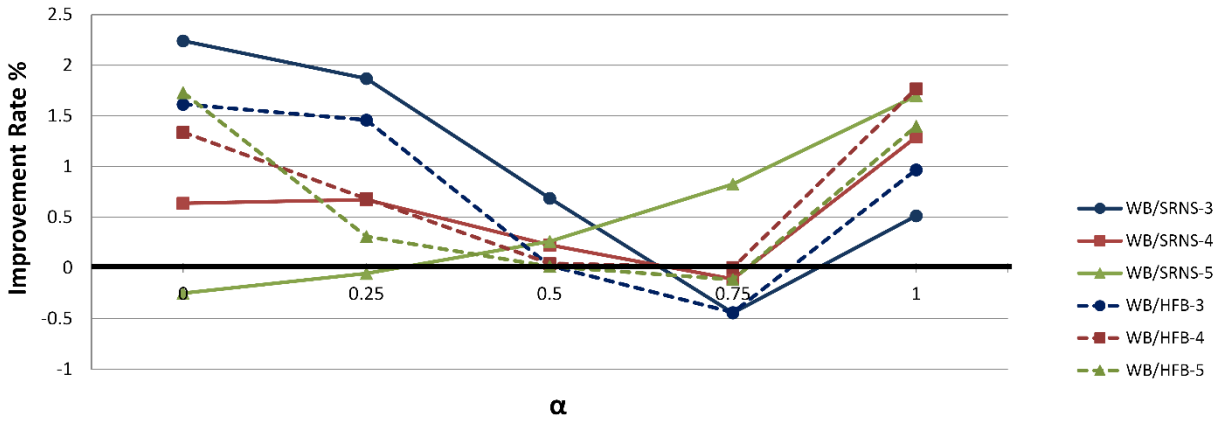


Figure 5-2 *The improvement achieved by WFB comparing with HFB and SRNS*

For WFB and HFB, the improvement rates of different *WIPs* are close to each other especially when the fixed task of W_2 is big. When the fixed task of W_1 is big, HFB has a worse performance than WFB and the differences in the performance shrink as *WIP* increases.

5.6 Conclusion

We investigated the performance of a rule which is a modified version of HFB. We call it WFB. The adjustment is done by dividing the full buffer by the fixed workload ratio instead of by half to get the cutoff value. We chose HFB among other rules whose cutoff value is calculated by halving of an amount of workload which is different from one rule to another since it shows the best or similar performance.

The new rule WFB presents the best performance comparing with HFB and SRNS in the most cases. However it indicate a weak performance with so tiny variance in scarce cases. In the light of these results, a rule considering the workload imbalance has a high potential to surpass the other rules for different workload configurations.

References

[1] Hopp, W. J.,Tekin, E., and Van Oyen, M. P., 2004. Benefits of Skill Chaining in Serial Production Lines with Cross-trained Workers. Management Science, 50(1): 83-98.

[2] Bartholdi III, J. J., D. D. Eisenstein. 1996. A production line that balances itself. Operations Research, 44(1) 21–34.

[3] KASMO, S., IKEDA, T., and ENKAWA, T., 2013. A Study on Elucidation of Complementary Effect between Helping Zone System and Introduction of Buffer Stations in Cell Manufacturing. *Journal of Japan Industrial Management Association*, 64(3): 472-479. (in Japanese)

[4] Askin, R.G., and Chen, J., 2006. Dynamic Task Assignment for Throughput Maximization with Work-sharing. European Journal of Operational Research, 168: 853– 869.

[5] Ostolaza, J., McClain, J. and Thomas, J., 1990. The use of dynamic (state-dependent) assembly-line balancing to improve throughput. Journal of Manufacturing and Operation Management, 3:105-133.

[6] Chen, J., and Askin, R.G., 2006. Throughput Maximization in Serial Production Lines with Work-sharing. International Journal of Production Economics, 99:88–101.

[7] McClain, J.O., Thomas, J., and Sox, C., 1992. On-the-fly Line Balancing with Very Little WIP. International Journal of Production Economics, 27: 283–289.

[8] Gel, E. S., Hopp, W. J. and Van Oyen, M. P., 2002. Factors affecting opportunity of worksharing as a dynamic line balancing Mechanism. IIE Transactions, 34(10): 847-863.

CHAPTER 6

CONCLUSION

Overview

This chapter sums up the findings and the derived insights from the experiments and investigations done in this research. The directions of future work are also stated.

6.1 General Summary

This work aims a scarcely targeted area in the work sharing under DLB mechanism, which is the workload. The workload in this research refers to the fixed workload since the shared work is managed by DLB. The shared work is managed dynamically based on the system status. On the other side, the fixed work is assigned at the line design stage, and this allocation continues working in this fixed structure. Therefore, investigating the effect of this factor can gives insights to the process designers to reduce any negative influence or boost any positive revenues on the performance.

This dissertation started by exploring the relation between DLB and Cell Manufacturing. DLB has a resembling concept in Japanese production system. The most similar notion is an divisional cell with a helping zone. The helping zone corresponds to the shared task in DLB. Since the idea of having a buffer between stations has started to be acceptable in cell manufacturing, we studied the effect of buffer in the cell under DLB as a method to manage the

helping zone. The findings of study shows that the variability plays the most vital factor to define the importance of having a buffer. A line with low variable processing times does not need a buffer and the work sharing can offset the effect of variability. Conversely. Introducing the buffer shows a remarkable improvement if the processing time is highly varied or the worker are not sufficiently cross-trained.

Several workload configurations are investigated. A measure of workload is made up to ease conducting the comparison between these configurations. The experiments have started by a so near optimal and easy to apply rule, SRNS (Smallest – R – No –Starvation). The performance of these configurations show a distinct pattern that is sensitive to the amount of available work in the line (*WIP*). The balanced case has the best performance, which is expected. For the other cases, with low *WIP* the big fixed task at the first station is more favorite while a small one is better when *WIP* is ample. In this sense, the ability of processing the job in a flexible order or breaking down the job to get a near balanced fixed workload is vital.

As any production line is prone to many distractions, our model was studied under several structural factors. Information accuracy, Granularity of shared task and variability are of the most important factors that have a notable influence on DLB performance. The information accuracy is an essential issue since the work sharing in DLB is managed based on the information about the system status. Two level of information accuracy have been tackled. Low information accuracy has a information about the available work in the a downstream buffer while high level needs additional information which is the amount of work at downstream station undone yet. We found that high level of information accuracy is not always good to the performance. The extreme cases of workload experience a better performance with the low level especially for low and high *WIP*. Besides, the case with a high level of information

accuracy does not indicate any advantage in case of exchanging the position of fixed tasks.

For the granularity, two points were treated, the order of subtasks when the granular shared task has different sized subtasks, and the workload balance. The first point shows an effect on the performance only when the shared task is large. In this case, processing the small subtask firstly gives a higher performance and the gain rises as WIP increases. The granularity affects the performance of workload in term of the value and pattern and its influence inflates as the size of shared task increases. The last factor is the variability. In this research, we took in account the variability of two types of task fixed and shared. Different combinations of variability were experimented. The findings indicate that the improvement efforts to reduce or eliminate the variability should focus on the shared task firstly to get a better performance. However, the effort should concentrate on the fixed task if it constitutes a big portion of job (i.e. bottleneck).

After conducting this intensive examination about the workload and its interactions with other factors, we finished this research by considering the workload in the cutoff rule used in DLB. The focus was on the ones that use the available amount of workload to get the cutoff value *R.* The rules mentioned in DLB literature divide the available work by 2 to get *R.* We came up with the idea of weighting the control rule by the ratio of fixed workload. We chose HFB as the best rule among the other rules in its category. The comparison between the new rule WFB, HFB and SRNS illustrates that in general WFB surpasses the other rules' performance. The improvement rate in some cases exceeds 2 % while for a very few cases the loss in performance is equal or less than 0.36%.

The findings of this research can give practical useful insights to the process designers and the production managers (especially assembly lines) where sharing work is applied. They

aid them to find out the most efficient design by manipulating the elements of process considering the work environment limitations. Moreover, the outcomes support the improvement efforts by specifying the most important areas to focus on firstly to augment the performance.

6.2 Limitations and Future Work

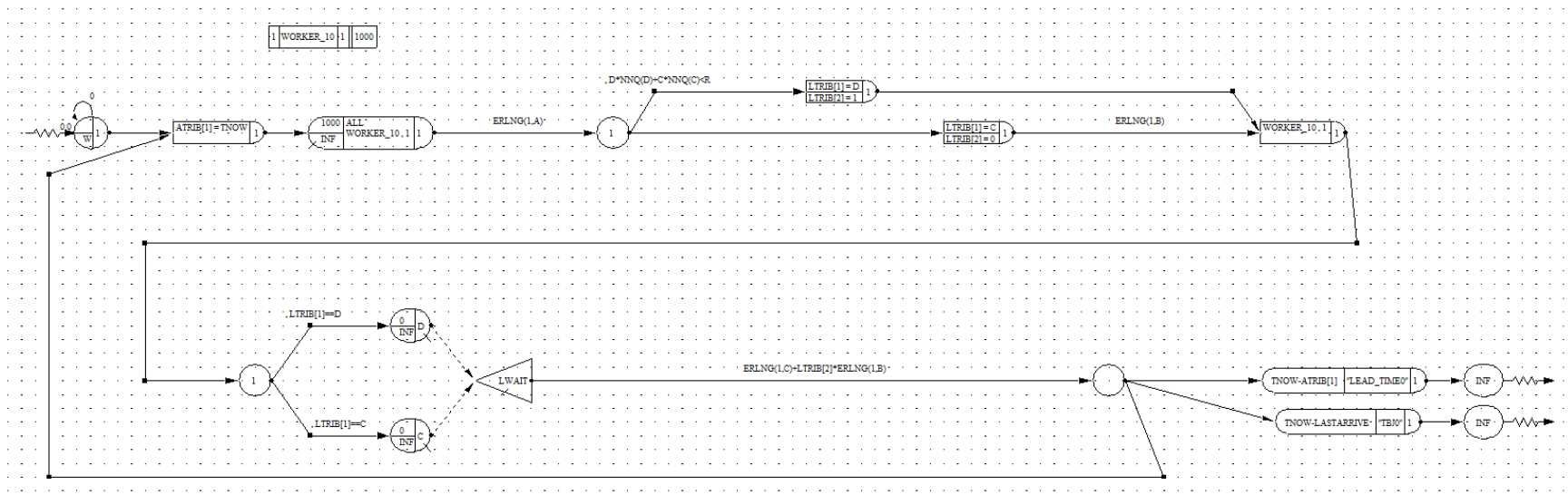
This research is the preliminary stage of exploring the effect workload under DLB as a mechanism of work sharing. Therefore, the model was a bit basic. However, extending the derived insights to more complicated cases is still possible. For longer lines than the one used here, if WIP is small, a line with big workload existed upstream will give a better performance than the one with downstream big workload. Another example, the average long-term performance of a line with more than one product to produce closely follows the same pattern of the most frequent product.

Nevertheless, further researches with more complicated model are important to reflect realistic situations. A line with more than one product and different bottleneck locations might be a interesting area to consider. The other directions of future work include representing the performance variances of workers when they process the shared task. The suggested rule is need a high level of information accuracy as in HFB, so finding a rule weighted by the workload ratio with low level of information accuracy is an attractive area to explore.

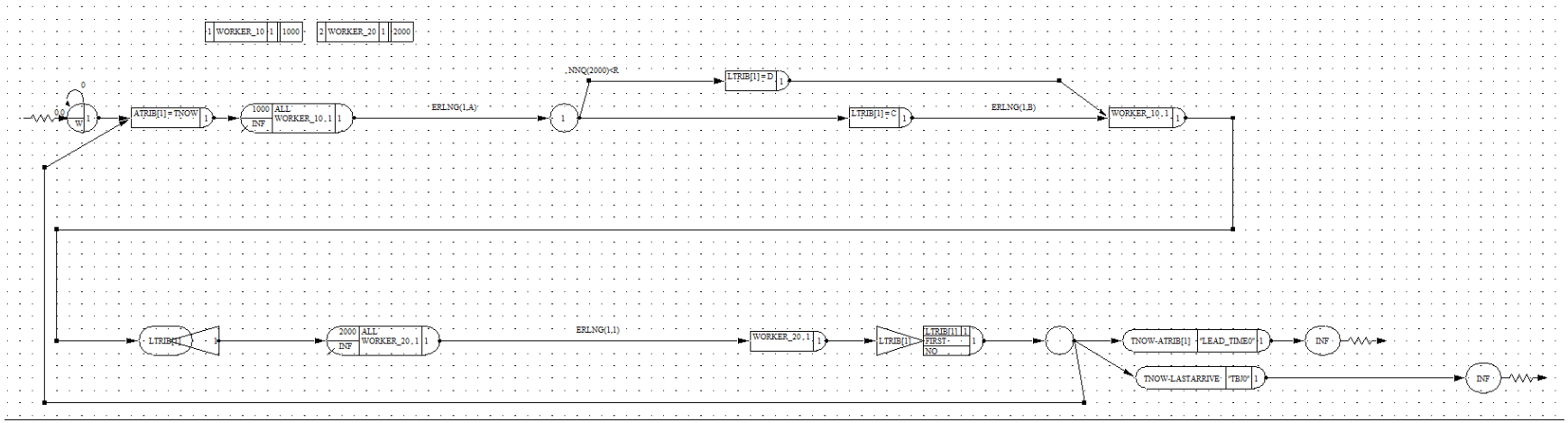
CONWIP (CONstatnt Work In Process) is used as a policy to control the number of jobs in the line. It is considered as a hyper policy (push-pull). Taking in account the pure push or pull system enriches the research in this area and gives important insights for such situations.

Appendix

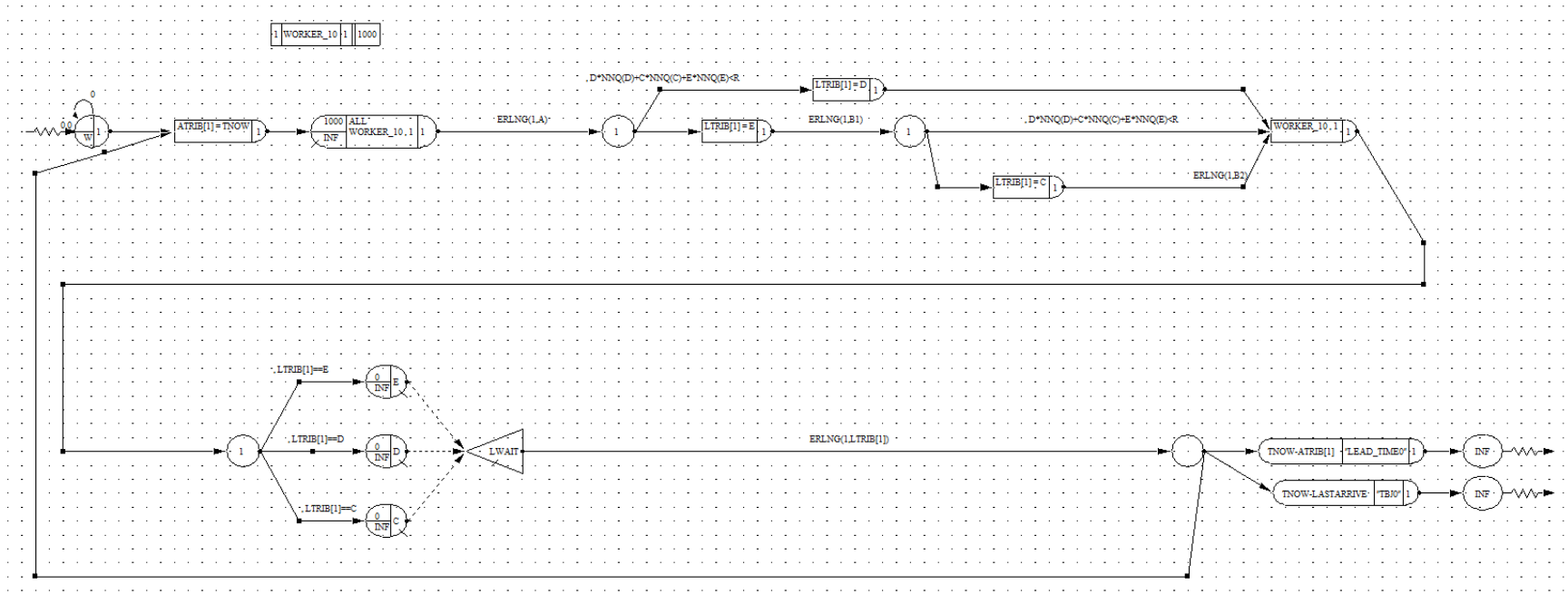
The Simulation Model of SRNS Rule



The Simulation Model of HFB Rule



The Simulation Model of SRNS Rule with Granularity



The Simulation Model of HFB Rule with Variability