國 立 交 通 大 學

資 訊 管 理 研 究 所

碩 士 論 文

二階段供應鏈之協同排程

Coordinated Scheduling in a Two-stage Supply Chain

研 究 生：謝欣穎

指導教授：林妙聰 博士

中華民國九十八年六月

二階段供應鏈之協同排程

Coordinated Scheduling in a Two-stage Supply Chain

研 究 生：謝欣穎	Student: Hsin-Ying Hsieh
指導教授：林妙聰	Advisor: Bertrand M.T. Lin

國立交通大學
資訊管理研究所
碩士論文

A Thesis
Submitted to the Institute of Information Management
College of Management
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master of Business Administration
in
Information Management
June 2009
Hsinchu, Taiwan, the Republic of China

中華民國九十八年六月

Title of Thesis: Coordinated Scheduling in a Two-stage Supply Chain
Name of Institute: Institute of Information Management
Student Name: Hsin-Ying Hsieh   Advisor Name: Professor Bertrand M.T. Lin

Abstract

In this thesis, we investigate a scheduling problem in concurrent open shops incorporating coordination among all participants. Weights are assigned to not only orders (jobs) but also product items (operations) to reflect the fact that any operations, parts or items need to be included in the formation of objective functions and the fact that the positions of buyers and sellers may not be equal in real-world application, such as buyers' market or sellers' market. Two integer programming formulations, based on positional variables and sequencing variables, of the studied problem are presented first. We then propose a three-phase algorithm, which comprises of a WSPT heuristic, the NEH algorithm and the variable neighborhood search, to produce approximate solutions to the computationally intractable problem. Finally, we will analyze the performance of the three-phase algorithm through the computational results.

Keywords: Concurrent open shops, total weighted completion time, coordination, supply chain, NEH algorithm; WSPT heuristic; variable neighborhood search.

摘要

在這篇論文中，我們將探討 Concurrent open shops 中協同合作的可能，並且不單只是考慮最小化完工時間總合，更加入權重的概念，做為最小化的目標式。因為不單單只是完工時間會對成本造成影響，應該要加入權重來詮釋 operations 或 orders 之間的相對重要性，以及反映現實市場中，可能存在買方市場或賣方市場這種買賣雙方地位不平等的情況。我們分別依據位置變數及相對位置變數提出兩個整數規劃模式來闡述這個題目。接著，我們提出一個由 WSPT 演算法、NEH 演算法與 variable neighborhood search 所構成的三階段近似解演算法，可以在短時間內為此計算複雜度極高的最佳化問題求得不錯的近似解。最後，我們進行實驗模擬並根據實驗結果，從多面向分析此三階段演算法的求解效益與效率。

關鍵字：Concurrent open shops、總加權完工時間、協同合作、供應鏈、NEH 演算法、WSPT 啟發式演算法、variable neighborhood search

Table of Contents

Abstract i
摘要 ii
Table of Contents iii
List of Figures iv
List of Tables iv
Chapter 1 Introduction 1
Chapter 2 Problem Statements 7
2.1 Definition and Related Work 7
2.2 Integer Programs 13
Chapter 3 Multi-Phase Approximation Algorithm 17
3.1 Multi-Phase Algorithm Construct 17
3.2 VNS-Based Algorithm 19
3.2.1 Initial solution 21
3.2.2 Neighborhood structure 21
3.2.3 Shaking 22
3.2.4 Local search 23
3.3 NEH-Based Algorithm 24
Chapter 4 Computational Experiments 27
4.1 Experiment Settings 27
4.2 Computational results and analysis 30
4.2.1 Solution Quality 30
4.2.2 Analysis of run time performance 32
Chapter 5 Concluding Remarks 50
Bibliography 52

List of Figures

Figure 1: The automobile production model 3

Figure 2: Construct of multi-phase algorithm 18

Figure 3: Pseudo-codes of variable neighborhood search (VNS).................................... 20

Figure 4: Pseudo-codes of Shaking... 22

Figure 5: Pseudo-codes of the NEH algorithm... 25

Figure 6: Experiment settings.. 29

Figure 7: Performance of heuristics in different settings 35

Figure 8: Correlations between α value and performance of dispatching rules 41

Figure 9: Correlations between α value and performance of NEH with different initial solutions 43

Figure 10: Correlations between α value and performance of NEH+VNS with different initial solutions 45

Figure 11: Correlations between α value and performance of VNS with different initial solutions 47

Figure 12: Correlations between data size and running time 49

List of Tables

Table 1: Weights and processing times of the operations 9

Table 2: Optimal (WSPT) sequences of operations on the machines 10

Table 3: Sequence of operations on all machines using the vendee's WSPT rule 11

Table 4: Optimal production schedule for the whole production system......................... 12

Table 5: Solution values and improvement ratios... 33

Table 6: Elapsed running time 34

Chapter 1

Introduction

The purpose of scheduling is to execute tasks by making use of the limited resources adequately in order to attain performance measures in an optimal way. In other words, scheduling is a decision-making process of allocating limited resources to tasks or activities along the time horizon and accomplish the optimization of specified objective(s). The resources may be capital, time, production lines, labor, etc. A task may be the production of components, the delivery of merchandize or services. The objective could be the minimization of the costs, the maximization of the profit, the minimization of completion times, etc. Effective and efficient scheduling can avoid unnecessary waste of resources, and thus enhance the efficiency of the projected operations.

In this thesis, we will investigate an operations scheduling problem that is further extended to the context of supply chain management. In the recent two decades, there has been a dramatic proliferation of research concerning supply chain management. Integration and coordination amongst the participants in the same supply chain or integration across different supply chains are especially noticed. Those topics focus on information sharing of productions, inventories and sales projections in order to enhance the efficiency and reduce the production cost of each participants in the supply chain. Each individual participant seeks to sketch an efficient but independent production plan achieving the lowest cost within his/her organization. While most of the literature on supply chain management treats only individual perspective, we intend to introduce the notion of a comprehensive perspective. There are sellers and buyers, no matter what kind of supply and demand relationship it is. Buyers always want to purchase economically, while sellers always expect to gain from bargains as much as possible. With this curb, individual optimization is limited, and one can only achieve the optimization for himself/herself at the cost of the benefit of others in the context of industrial networks. Optimization of every individual is hard to carry out, since no one is willing to be the victim. Therefore, we discuss the possibility of mutual benefits and provide a practical solution to promote the concept of working cooperatively, instead of being rivals that could reduce the system-wide benefit of every participant in the supply chain. While someone might lose when enriching the profit of the entire system, they can get damages from the margin, and then, the remainder margin will be divided up among the participants in the supply chain system. In this thesis, the subject we studied is the production model including one buyer and several manufacturers. To elaborate on the idea clearly, we provide an automobile production model as an example.

Suppose there is an automobile company which sells customized cars. **Figure 1** shows the model of this production system. The three components which can be customized are engine, wheel, and panel. The automobile company receive five orders

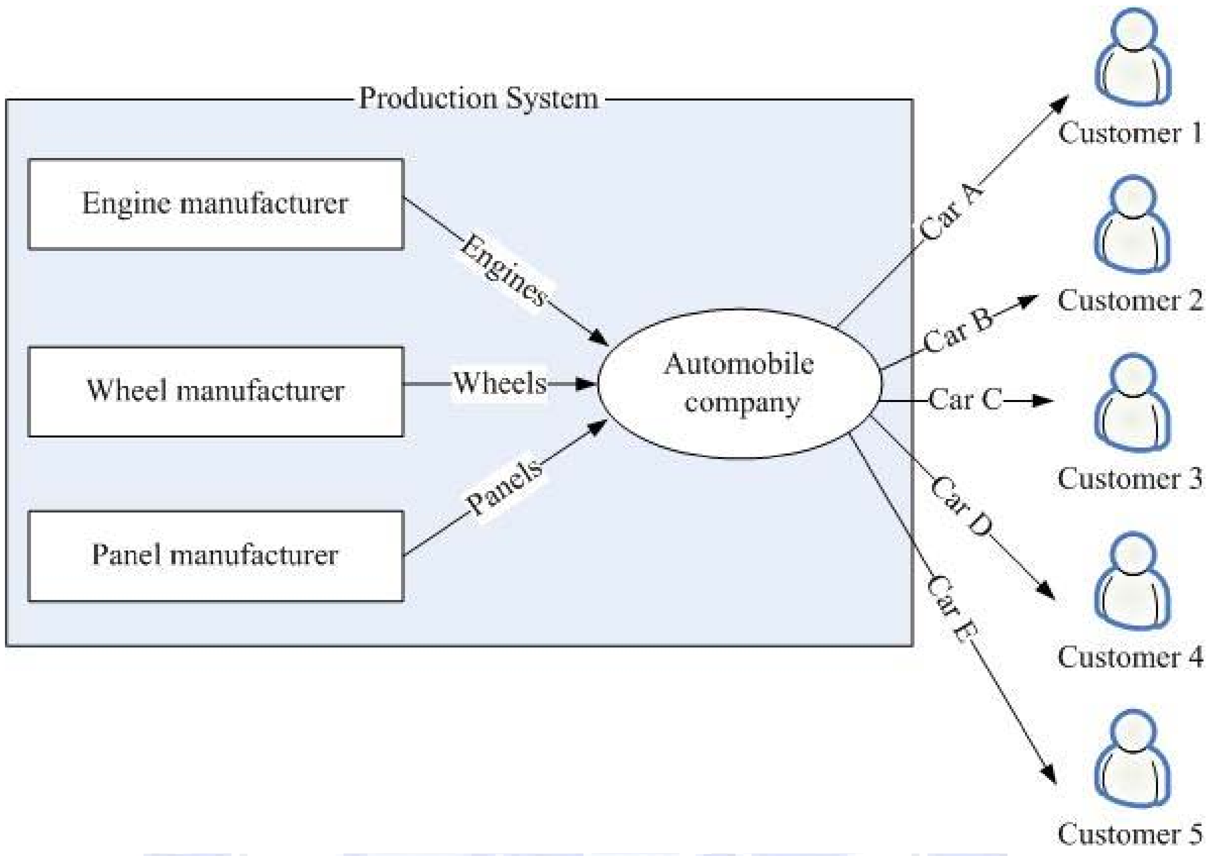


Figure 1: The automobile production model.

from five different customers at the same time, and then it place five orders to the engine manufacturer, the wheel manufacturer, and the panel manufacturer, respectively, according to the specific requests of each customer (5 orders and 3 manufacturers). The components with different specifications would cause the differences in processing times, selling prices, and even the profits that the automobile company or the three manufacturers might earn. For the three manufacturers, they intend to plan an efficient schedule which is most advantageous for them to produce the product items. However, the automobile company has to wait until all of the three components ordered by the same customer are finished before they can fulfill this order. We assume that there is a fixed standard operation time for the delivery between manufacturers and automobile company, and a standard procedure for the assembly. Furthermore, the delivery and assembly are never changed even if different schedule policies are applied; besides, they will not make any difference on the final result. For this reason, we will ignore the processing times and expenses on delivery and assembly; in other words, a car is finished as long as all of its three parts are produced and available.

On the automobile company side, they expect the three components of the same car are finished at the same time, or the company has to hold the finished component(s) until all of them are finished. It may cause some costs, such as that for inventory. Besides, the orders may have different priorities, the relatively important orders are supposed to be finished before those with relatively low importance. The reason for the priority differences might be that a VIP customer has a higher priority than regular customers, or the customer requests a specific delivery time, and many others. Though it is very unlikely to charge a penalty for a delayed delivery, customer satisfaction will probably dwindle. The intangible cost like this may substantially hurt the image of enterprises. Therefore, the automobile company will prioritize the components during production planning.

It is clear that the standpoints are quite different between the car company and the three manufacturers. The car company is going to get into trouble, if production schedule is arranged in the light of manufacturers' preferences. For one thing, the partially fulfilled orders with finished components need to be disposed, for another, the car company has to deal with the customers who are impatient after waiting for a long time. For these reasons, it is too inefficient for the automobile company to accept this scheduling policy. On the contrary, if the schedule follows the company's prioritization, the profit of some, or even all of the manufactures will be sacrificed. The conflicts between buyers and sellers are quite commonplace in a supply chain. In this thesis, we are considering if there is a way to reconcile buyers and sellers, and lead them to make decisions by the consensus that the production schedule will benefit all of them. More specifically, the destination of this studied problem is the minimization of system-wide cost for the automobile production. To discuss the model with one buyer and several sellers generally, the term "vendee" is used to describe the buyer which is the automobile company in this example. Also, the term "manufacturer" is used to describe the roles who produce components, that is the engine manufacturer, the wheel manufacturer, and the panel manufacturer in this example. Besides, we defined a single component production process as an operation in the rest of this thesis. The model of the supply and demand relationship mentioned above is exactly equal to the $PD||\sum_i w_iC_i + \sum_k \sum_i w_{ik}C_{ik}$ problem in scheduling domain.

The rest of this thesis is organized as follows. In Chapter 2, we will define the studied problem in formal mathematical statements. Two integer programming formulations, based upon positional variables and sequencing variables, will be proposed. As will be explained in Chapter 2, the studied problem is strongly NP-hard, indicating that it is very unlikely to design an efficient solution algorithm. In Chapter 3, we will propose a three-phase heuristic to tackle this computationally hard problem. The three-phase heuristic consists of two dispatching rules, an NEH-based algorithm and a VNS algorithm. The computational experiments and analysis on the proposed algorithms will be presented in Chapter 4. In Chapter 5, we present the concluding remarks and suggest potential topics for further research.

Chapter 2

Problem Statements

In the problem setting, if we consider only the optimization of the vendee's perspective, then the problem reduces to the so-called concurrent open shop scheduling, or order scheduling. We start this chapter with the introduction of order scheduling, followed by two integer programming (IP) formulations. Then, an example will be given to illustrate the problem definition. Related work will be also reviewed.

2.1 Definition and Related Work

Ahmadi and Bagchi (1990) could be the first work on concurrent open shop scheduling. Lee et al. (1993) and Potts et al. (1995) investigated two-stage assembly flowshops, which is a generalization of concurrent open shop scheduling by including a second-stage assembly machine. The term "concurrent open shops with job overlaps" was coined by Wagneur and Sriskandarajah (1993). Wang and Cheng (2003) studied the same setting from another view point, called order scheduling. An order is a demand of a certain commodity including several items. There is no relation between items, so they can be produced in an arbitrary order on their dedicated machines. Simultaneous processing is allowed in concurrent open shops. An order is completed only if all of its items are finished. In the automobile example, an automobile consists of several components, such as engine, panel, and wheels, which are called items. Items would be produced on their dedicated production lines, and an automobile can be assembled until all of its components are manufactured and available, i.e. an order is complete.

We define the problem with formal statements. There are n orders $O = \{O_1, O_2, \ldots, O_n\}$ to process, and each order O_i is composed of m operations $O_{ki}, 1 \leq k \leq m$, to be processed on m independent dedicated machines $M_1, M_2, \ldots, M_m$. Let O_{ki} stand for the k^{th} operation of order O_i which can be processed on only machine M_k. Operations of the same order are independent with no precedence constraints among them; in other words, they can be processed in an arbitrary order, and are allowed to be processed on their dedicated machines simultaneously. Each machine can process only one operation at a time and no preemption is allowed. For this production model, the unique characteristic different from open shop is that the operations of the same order can be processed simultaneously.

The notations used throughout this thesis are introduced as follows. C_{ki} denotes the completion time of operation O_{ki} on machine M_k, and each O_{ki} has a weight w_{ki}. The completion time of order O_i is defined as $C_i = \max_{1 \leq k \leq m}\{C_{ki}\}$, since each order is completed when all of its operations are finished. Each order O_i is also associated with a weight w_i to reflect the relative importance of this order. Let p_{ki} be the processing time of operation O_{ki}. The binary decision variable U_{kil} is equal to 1 if operation O_{ki}

is scheduled at position l on machine M_k, and 0 otherwise. Another binary variable y_{kij} is set to be 1 if the completion of operation O_{kj} does not succeed operation O_{ki} on machine k, and 0 otherwise. In this study, we consider not only the total sum of the weighted completion times of mn operations on all m machines, but also the sum of the weighted completion times of n orders. In the studied problem, the objective function is defined as the minimization of the sum of the weighted completion times of mn operations and the weighted completion time of n orders.

An instance with 5 orders and 3 manufacturers is given in **Table 1** to illustrate the problem setting.

Table 1: Weights and processing times of the operations.

Order	O_1	O_2	O_3	O_4	O_5
w_i	14	12	20	16	21
p_{1i}	69	62	80	35	25
w_{1i}	9	1	6	5	9
p_{2i}	36	16	92	8	72
w_{2i}	5	1	7	1	2
p_{3i}	37	30	22	50	65
w_{3i}	1	10	7	10	10

Due to the selfishness of each manufacturer/machine, he/she will arrange the operations based upon his/her own optimal policy, which can be achieved using the wellknown **WSPT** (weighted shortest processing time first) rule. The optimal schedules on the machines are given in **Table 2**.

Table 2: Optimal (WSPT) sequences of operations on the machines.

M_1	O_{15}	O_{14}	O_{11}	O_{13}	O_{12}
M_2	O_{21}	O_{24}	O_{23}	O_{22}	O_{25}
M_3	O_{32}	O_{33}	O_{34}	O_{35}	O_{31}

Under this scheduling strategy, the WSPT rule produces the minimum weighted sum of the completion times of mn operations, i.e. $\sum_k \sum_i w_{ik}C_{ik}$ =8,545. Subject to the three sequences, the weighted sum of the completion times of n orders is rather high $\sum_i w_iC_i$ =16,828. The system-wide cost is thus $\sum_i w_iC_i + \sum_k \sum_i w_{ik}C_{ik} = 25{,}373$.

In contrast to the scheduling policy which only takes the manufacturer's preference into account, the decision of the vendee is rather hard to optimize. If only $\sum_i w_iC_i$ is considered, implying the absolute dominance of the vendee in the system that he/she determines the production schedules over all machines, then the problem becomes the so-called order scheduling or concurrent open shop scheduling, denoted by $PD||\sum_i w_iC_i$, where PD dictates the characteristic of parallel dedicated machines. It has been shown in the literature that it suffices to consider only the scenarios where all machines have the same processing sequence and that the concurrent open shop scheduling of minimizing the weighted sum of completion times remains strongly NP-hard even if there are only two machines and all orders are equally weighted (Roemer, 2006). Therefore, it is very unlikely to design an efficient algorithm for solving the vendee's problem. Intuition suggests that the vendee can apply a simple *greedy* dispatching rule to decrease his/her cost, although the optimal solution for the vendee ($\sum_i w_i C_i$) is not guaranteed. According to this dispatching rule, the orders are scheduled in non-increasing order of $w_i / \max_{1 \leq k \leq m}\{p_{ki}\}$, namely, the sequence of each machine is the same as others. The schedule is shown in **Table 3**.

Table 3: Sequence of operations on all machines using the vendee's WSPT rule.

M_1	O_{14}	O_{15}	O_{13}	O_{12}	O_{11}
M_2	O_{24}	O_{25}	O_{23}	O_{22}	O_{21}
M_3	O_{34}	O_{35}	O_{33}	O_{32}	O_{31}

As we expected, the sum of the weighted completion times of the n orders is reduced by 3,786, while the sum of the weighted completion times of mn operations $\sum_k \sum_i w_{ik} C_{ik}$ =11,166, showing an increase of 2,621. In this manner, the system-wide cost $\sum_i w_i C_i + \sum_k \sum_i w_{ik} C_{ik}$ reduces to 24,208.

This is merely a case that considers only the vendee rather than the manufacturers when planning a schedule gets a better result. Judging from the above, we can find that there seems to be a contradiction between the profits of the vendee and the m manufacturers, that is, the decrease in the vendee's expense would cause the increase in the total cost of the m manufacturers, and vice versa. Is it possible to transfer the relationship between the two from opposing position to complementary, in order to enhance the profit for each side? There is still room for improvement, if all individuals in this supply chain can actually act in concert and devise a multi-lateral policy of the production schedule. The ideal production schedule is shown in **Table 4**.

Table 4: Optimal production schedule for the whole production system.

M_1	O_{15}	O_{14}	O_{13}	O_{11}	O_{12}
M_2	O_{24}	O_{23}	O_{25}	O_{21}	O_{22}
M_3	O_{34}	O_{32}	O_{33}	O_{35}	O_{31}

Under this schedule, the system-wide cost ($\sum_i w_iC_i + \sum_k \sum_i w_{ik}C_{ik}$) declines more to 23,480, which consists of the sum of the weighted completion times of n orders ($\sum_i w_iC_i$=13,759) and the sum of the weighted completion times of mn operations ($\sum_k \sum_i w_{ik}C_{ik} = 9,721$). Neither the vendee nor the manufacturers get the best outcome they have looked for, but this is the optimal solution for the whole supply chain. The intrinsic nature of the ideal schedule is to reach for a consensus that the vendee or some of the manufacturers needs to make concessions in order to enhance the benefit of the supply chain. Since the $PD||\sum_i w_iC_i$ problem is strongly NP-hard, the $PD||\sum_i w_iC_i + \sum_k \sum_i w_{ik}C_{ik}$ problem of interest in this thesis is also hard to solve. Hence, how many concessions should be made by each one of them is the crux in this thesis. However, design of schemes to compensate the members for the loss at this optimal system-wide production schedule is not addressed in this thesis.

A number of studies have been conducted using the total completion times or weighted total completion times of orders as a research topic. For example, in Cheng et al. (1997), the objective function depends on the delivery times of batches. Customer order scheduling (Leung et al. 2007, Lin and Kononov 2007, Wang and Cheng 2007) and concurrent open shop scheduling (Roemer 2006) are three examples focused the aggregation concept in the order scheduling area. The majority of research in order scheduling has focused on the aggregation concept. There has been far less research on the concept of disaggregation which means each item in an order has its own identity, thus the completion time of items is always ignored. Especially, Chen and Hall (2007) concerned about both aggregation and disaggregation. However, the relative importance between items or orders is not considered in their study. Up to this point, there could be no research studying this issue. Therefore, we will focus on the order scheduling in concurrent open shop to consider both of the aggregation and disaggregation concept with weighted total completion times in this thesis.

2.2 Integer Programs

To further describe the problem setting, we propose two integer programming formulations, based upon positional variables and sequencing variables, respectively. The models provide potential use in the pursuit of either exact solutions or approximate solutions. We first introduce the model (**IP-Pos**) described in positional variables. Recall that binary decision variable u_{kil} is equal to 1 if operation O_{ki} is scheduled at position l on machine M_k; and 0, otherwise. To facilitate the presentation, we introduce auxiliary binary variables y_{kij} to indicate whether operation O_{ki} precedes operation O_{kj} on machine M_k or not.

(**IP-Pos**) Minimize $\sum_i w_i C_i + \sum_k \sum_i w_{ik} C_{ik}$

subject to

$$
\begin{array}{lll}
C_{ki} = \sum_j p_{kj} y_{kij} & \forall k, i & (1) \\
(y_{kij} - \sum_{l'=1}^{l} u_{kjl'}) u_{kil} = 0 & \forall k, i, l & (2) \\
\sum_l u_{kil} = 1 & \forall k, i & (3) \\
\sum_i u_{kil} = 1 & \forall k, l & (4) \\
C_i - C_{ki} \geq 0 & \forall k, i & (5) \\
y_{kij} \in \{0, 1\} & \forall k, i, j & (6) \\
u_{kil} \in \{0, 1\} & \forall k, i, l & (7) \\
1 \leq k \leq m & k : \text{machine index} & \\
1 \leq i \leq n & i : \text{order index} & \\
1 \leq l \leq n & l : \text{position index} &
\end{array}
$$

Constraints (1) ensure that the completion time of operation O_{ki} is the sum of the processing time of all the operations that have been sequenced not after operation O_{ki} on machine M_k. Constraints (2) give auxiliary equations to clarify the relative positions of the operations on their dedicated machines. Constraints (3) and (4) guarantee that each operation can be processed exactly once, and each machine can only process one operation at any time. Constraints (5) state that an order is completed only if all of its operations are finished on their dedicated machines. Constraints (6) and (7) confine the variables to be binary. Variables k, i, and l are machine index, order index, and positional index, respectively. In the proposed model, $O(mn^2)$ variables and $O(mn)$ constraints are involved.

Note that the (**IP-Pos**) model is not linear due to the quadratic terms in constraints (2). In the second model, we instead use the sequencing model to circumvent the difficulty in dealing with non-linear constraints. In model (**IP-Seq**), binary variables y_{kij} are now the decision variables rather than auxiliary.

(**IP-Seq**) Minimize $\sum_i w_i C_i + \sum_k \sum_i w_{ik} C_{ik}$

subject to

$$C_{ki} = p_{ki} + \textstyle\sum_j p_{kj} y_{kij} \qquad \forall k, i \qquad (8)$$

$$C_{ki} - C_{kj} - p_{ki} + (1 - y_{kij})M \geq 0 \qquad \forall k, i, \forall j > i \qquad (9)$$

$$C_{kj} - C_{ki} - p_{kj} + (1 - y_{kji})M \geq 0 \qquad \forall k, i, \forall j > i \qquad (10)$$

$$y_{kij} + y_{kji} = 1 \qquad \forall k, i, \forall j \neq i \qquad (11)$$

$$C_i - C_{ki} \geq 0 \qquad \forall k, i \qquad (12)$$

$$y_{kij} \in \{0, 1\} \qquad \forall k, i, \forall j \neq i \qquad (13)$$

$$1 \leq k \leq m \qquad k : \text{machine index}$$

$$1 \leq i \leq n \qquad i : \text{order index}$$

$$1 \leq j \leq n \qquad j : \text{order index}$$

Constraints (8) relate that the completion time of operation O_{ki} consists of its processing time and the sum of the processing times of all its preceding operations on machine M_k. Constraints (9) and (10) specify that only one job can be processed at a time, i.e., operation O_{ki} is sequenced either before operation O_{kj} or after operation O_{kj}. Constraints (11) dictate that both operations O_{ki} and O_{kj} have to be sequenced, i.e. either $y_{kij} = 1$ or $y_{kji} = 1$. Constraints (12) state that an order is completed only if all of its operations are finished. Binary constraints are given in (13). Variable k is machine index, and variable i and j are both order indexes. Similarly, $O(mn^2)$ binary variables are used and $O(mn)$ constraints are defined. One feature of this model is due to the linearity. Nevertheless, the use of big-M in constraints (9) and (10) may diminish the efficiency of linear models.

Chapter 3

Multi-Phase Approximation Algorithm

In this chapter, we present a multi-phase algorithm for solving the coordinated scheduling problem. The construct of the algorithm will be presented first and followed by the elaboration of the two WSPT-based heuristics, a VNS-based algorithm and an NEH-based algorithm.

3.1 Multi-Phase Algorithm Construct

The proposed algorithm consists of three phases so as to minimize the system-wide cost. Phase one of the algorithm seeks to derive initial solutions by two different WSPT-based heuristics. The first heuristic generates two initial schedules according to the WSPT rule, and the $\text{WSPT}_{\max}$ rule. In phase two, we use the two initial solutions to start a VNS-based algorithm and an NEH-based algorithm, respectively. In phase three, we take the solution produced by the NEH-based algorithm as an initial solution and invoke the VNS-based algorithm. Including the initial solutions, there are eight different solutions we will obtain through the above three phases. The multi-phase algorithm construct is depicted in **Figure 2**.

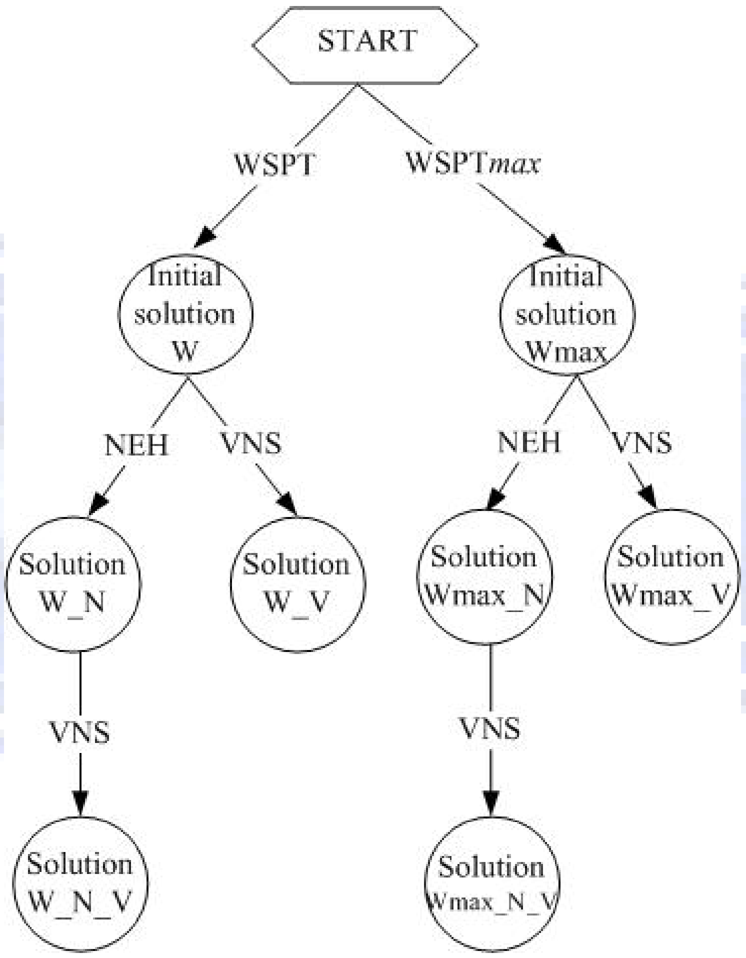


Figure 2: Construct of multi-phase algorithm.

3.2 VNS-Based Algorithm

The variable neighborhood search (VNS) algorithm was proposed by Mladenovic and Hansen (1997). It systematically changes the neighborhood within a possibly randomized local search, and yields a simple and effective meta-heuristic for combinatorial optimization problems. Pseudo-codes of the VNS is given in **Figure 3**. In the procedure, $\mathcal{N}_z$ $(z = 1, \ldots, z_{\max})$ denotes a finite set of pre-selected neighborhood structures, and $\mathcal{N}_z(s)$ of solution s is the set of solutions in the z^{th} neighborhood of s. To apply the VNS algorithm to an optimization problem, four components have to be specified:

1. A mechanism for generating an initial solution;
2. A well-defined neighborhood structure $\mathcal{N}_z$ that decides to which the Shaking mechanism is applied;
3. A procedure called Shaking that disturbs the incumbent solution sequence according to the z^{th} neighborhood structure; and
4. A procedure LocalSearch that moves from solution s' to a local optimum s''.

The VNS differs from other local search heuristics in neighborhood structures. Most local search heuristics use only one fixed neighborhood structure, i.e., $z_{\max} = 1$. Moreover, the solution-shaking mechanism which corresponds to disturbance in the search space is applied to permit possible escape from local optimum.

```
PROCEDURE Variable Neighborhood Search
Begin
Generate initial solution $s$;
Select the set of neighborhood structures $\mathcal{N}_z$, $1 \leq z \leq z_{\max}$;
Determine the stopping criteria;
Set $z := 1$;
Repeat
    $s' := Shaking(s, z)$;
    $s'' := LocalSearch(s')$;
    If $s'' \succ s$ then $s := s''$; $z := 1$;
    else $z := z + 1$;
Until $z = z_{\max}$ or stopping criterion met
end
```

Figure 3: Pseudo-codes of variable neighborhood search (VNS).

To apply the VNS to the studied problem, we need to (1) define the **neighborhood structure** $\mathcal{N}_z$; (2) establish the **Shaking** and **LocalSearch** mechanisms; and (3) generate an **initial solution**. We will discuss the details in the following. The last but not the least, the stopping criteria is set to be a specified maximum number of iterations which is set to be four in our experiments.

3.2.1 Initial solution

We use two dispatching rules to construct two possibly different initial solutions. The first one is the WSPT rule that arranges the operations in non-increasing order of w_{ki}/p_{ki} on the corresponding dedicated machine. Individual machines may have distinct sequences in the WSPT arrangement. Another heuristic is due to a simple greed dispatching rule $\text{WSPT}_{\max}$ which sorts the orders in non-increasing order of $w_i/\max_{1\leq k\leq m}\{p_{ki}\}$. All dedicated machines follow a common processing sequence.

3.2.2 Neighborhood structure

The solution space is denoted by $\mathscr{S}$, and function $\rho(s,s')$ gives the distance between solutions s and s'. Here, the distance means the number of manipulatable operations, like forward/backward insertion and swap, applied to one solution so as to reach the other one. If the difference between schedule s and schedule s' can be categorized in the z^{th} structure, then $\rho(s,s')=z$. The neighborhood structure characterized by solution $s\in\mathscr{S}$ and positive integer $z\geq 1$ is given as

$$\mathscr{N}_z(s)=\{s'|\rho(s,s')=z\}.$$

When implementing the neighborhood structures, we use the reverse function to generate the schedule s'. In this heuristic, we set the $z_{\max}=3$, which means there are three kinds of neighborhood structures. The structures will be defined in the first place. If a schedule s can be transformed into another schedule s' through a single insertion-move, then $\rho(s,s')=1$. That means s' will be derived from s with an insertion-move if $z=1$, i.e. $s'=$ *insertion-move*(s); so that neighborhood structure $z=1$ is constructed.

Similarly, when $z = 2$ and $z = 3$, 2-opt and swap-move will be applied respectively to transform s into s', i.e.,$s' = 2\text{-}opt(s)$, if $z = 2$; $s' = swap\text{-}move(s)$, if $z = 3$. All of the three structures are constructed. Details of these operations will be described in the next section.

3.2.3 Shaking

The shaking procedure will manipulate the neighborhood structure we defined earlier. It will perturb sequence s according to the neighborhood structure z. The shaking procedure is shown in **Figure 4**. The operations or moves are defined as follows.

```
PROCEDURE Shaking
Begin
    If $z = 1$ then $s' := insertion\text{-}move(s)$;
    If $z = 2$ then $s' := 2\text{-}opt(s)$;
    If $z = 3$ then $s' := swap\text{-}move(s)$;
end
```

Figure 4: Pseudo-codes of Shaking.

Insertion-move

Let $\pi_k = (\pi_{k,1}, \pi_{k,2}, \ldots, \pi_{k,n})$ denote a sequence on machine M_k. Let (i, j) be a pair of positions from the sequence. A new permutation π'_k is obtained by removing job $\pi_{k,i}$ at position i and inserting it at position j. If $i < j$ we obtain $\pi'_k =$ $(\pi_{k,1}, \ldots, \pi_{k,i-1}, \pi_{k,i+1}, \ldots, \pi_{k,j}, \pi_{k,i}, \pi_{k,j+1}, \ldots, \pi_{k,n})$ and if $i > j$ we get $\pi'_k = (\pi_{k,1}, \ldots, \pi_{k,j-1}, \pi_{k,i}, \pi_{k,j}, \ldots$ We arbitrarily choose a pair of positions from sequence s to perform the insertion-move procedure on each machine and repeat this process x times for $\mathscr{N}_1$, where $1 \leq x \leq 10$.

2-OPT

Let (i, j) be a pair of positions. From sequence π, the new permutation π'_k is obtained by reversing the subsequence $(\pi_{k,i}, \pi_{k,i+1}, \ldots, \pi_{k,j-1}, \pi_{kj})$ of π into $(\pi_{k,j}, \pi_{k,j-1}, \ldots, \pi_{k,i+1}, \pi_{k,i})$ so that we get $\pi'_k = (\pi_{k,1}, \ldots, \pi_{k,j}, \pi_{k,j-1}, \pi_{k,j-2}, \ldots, \pi_{k,i+2}, \pi_{k,i+1}, \pi_{k,i}, \ldots, \pi_{k,n})$. We arbitrarily choose a pair of positions in sequence s to perform the 2-OPT move on each machine and repeat this process x times for $\mathscr{N}_2$, where $1 \leq x \leq 10$.

Swap-move

A sequence $(\pi_{k,1}, \ldots, \pi_{k,i}, \pi_{k,i+1}, \ldots, \pi_{k,n})$ can be transformed to $(\pi_{k,1}, \ldots, \pi_{k,i+1}, \pi_{k,i}, \ldots, \pi_{k,n})$ through a swap-move of two consecutive positions. In this heuristic, we apply $n/2$ swap-moves to all consecutive odd-even positions in $\pi_k = (\pi_{k,1}, \pi_{k,2}, \pi_{k,3}, \pi_{k,4}, \ldots, \pi_{k,n-3}, \pi_{k,n-2}, \pi_{k,n-1}, \pi_{k,n})$ and obtain $\pi'_k = (\pi_{k,2}, \pi_{k,1}, \pi_{k,4}, \pi_{k,3}, \ldots, \pi_{k,n-2}, \pi_{k,n-3}, \pi_{k,n}, \pi_{k,n-1})$.

3.2.4 Local search

Interchange-move is applied to our local search procedure. In an interchange-move, we select two positions that are not necessarily consecutive, and then swap their positions. The neighborhood size of a given solution is thus $n(n-1)/2$, i.e. the number of possible moves for a sequence is $O(n^2)$. From among the $O(n^2)$ neighbors, we select the best one for improvement. If the best neighbor has an objective value larger than the current one, then it is discarded and no move is made. In the studied problem with m machines, we perform local search on one sequence of a machine at a time. We identify the machine with the largest weighted total completion times ($\sum_{i=1}^{n} w_{ki}C_{ki}$), called *bottleneck machine*, to perform the local search, the sequence on other else machines remain unchanged. Next, we find the bottleneck machine from the remaining $m-1$ untouched machines to perform the local search. The process is repeated until the local search has been performed on all machines. Each machine performs the local search once and only once, so that the overall running time of this heuristic is $O(mn^2)$.

3.3 NEH-Based Algorithm

The NEH algorithm was proposed by Nawaz, Enscore and Ham(1983) for flowshop scheduling to minimize the makespan, i.e. the maximum complexion time. It has been widely applied to flowshop scheduling with different objective functions and impressive performances have been reported. The success due to not only the simplicity but also the effective arrangement of jobs. The concept of this algorithm is to generate a sequence by successively arranging a relatively optimal position for each job, so that a local optimal sequence can be generated. The pseudo-codes of the NEH algorithm are given in **Figure 5**. The insertion move, which we have introduced in the previous section, is used to locate the best position when a job is considered for insertion into the current partial sequence.

```
PROCEDURE NEH
    Generate initial solution $s_0$;
    $s' := s_0$; $s'' := s_0$;
    For $i := 1; i \leq n-1; i++$ do
    Begin
        $BestPosition := i$;
        For $j := i+1; j \leq n; j++$ do
        Begin
            $s' := Insertion\text{-}move(s', i, j)$;
            If $s' \succ s''$ then
            Begin
                $s'' := s'$;
                $BestPosition := j$;
                $s' := Recover(s', i, j)$;
            end
            else $s' := Recover(s', i, j)$;
            $s' := Insertion\text{-}move(s', i, BestPosition)$;
            $s'' := s'$;
        end
    end
```

Figure 5: Pseudo-codes of the NEH algorithm.

Take a single sequence positioning for example. Let $J_1, J_2, J_3, J_4, J_5, J_6, J_7$ be the initial sequence, in which job J_1 is at first position, job J_2 is at second position and so forth. For J_1 there is only one position possible, so we have $(J_1, \| J_2, J_3, J_4, J_5, J_6, J_7)$, where $\|$ is a flag indicating the job being considered. For J_2, there are two possible positions to choose from, before or after J_1, if J_2 executes after J_1, then the objective value would become smaller and J_2 is going to be positioned right after J_1, and vice versa. The sequence will therefore be $(J_1, J_2, \| J_3, J_4, J_5, J_6, J_7)$. Similarly, that for J_3 there are three positions, before both J_1 and J_2, between J_1 and J_2, or after both J_1 and J_2. If the optimal position, which would achieve the objective value smallest, for J_3 is before both J_1 and J_2, then the sequence will be selected $(J_3, J_1, J_2, \| J_4, J_5, J_6, J_7)$. According to the above sequence positioning, one can fine a local optimal order for the sequence on a specific machine. The NEH algorithm applied to a single machine requires $O(n^2)$ running time. To apply the NEH algorithm to the problem with m machines, we also follow the bottleneck-machine-first rule which is related to the notion previously mentioned. As the process will be repeated for m machines, the overall running time of the NEH-based heuristic for improving an initial solution is $O(mn^2)$.

Chapter 4

Computational Experiments

In this chapter, we conduct computational experiments to study the performance of the proposed algorithms. We will first introduce the platform and parameters setting of the experiments, and then present the analysis of the numerical results of the experiments.

4.1 Experiment Settings

We designed a series of computational experiments. The algorithms were implemented in C++ and tested on a personal computer with a Pentium D 2.8 GHz CPU and 1GB memory running under Microsoft Windows XP.

For certain values of n and m, we computed the heuristic solutions for test instances, which were randomly generated as follows:

- The number of orders n is in $\{50, 100\}$.
- The number of machines/manufacturers m is in $\{3, 5, 7\}$.

- The processing times p_{ki} were independently drawn from a uniform integer distribution over the interval $[1, 100]$.

- The weights w_{ki} associated with each operation of various products were selected from the interval $[1, 10]$.

- The weight of order O_i is defined by constant $\alpha \in \{1, 0.5, 1/m\}$ such that $w_i = \alpha \times \sum_{k=1}^{m} w_{ki}$.

According to the parameters n, m, and α, there are 18 combinations to be examined in the whole setting. For each combination of parameter values of n, m, and α, we deploy the proposed three-phase heuristic to solve 10 test instances. A chart of the experiment setting is depicted in **Figure 6**.

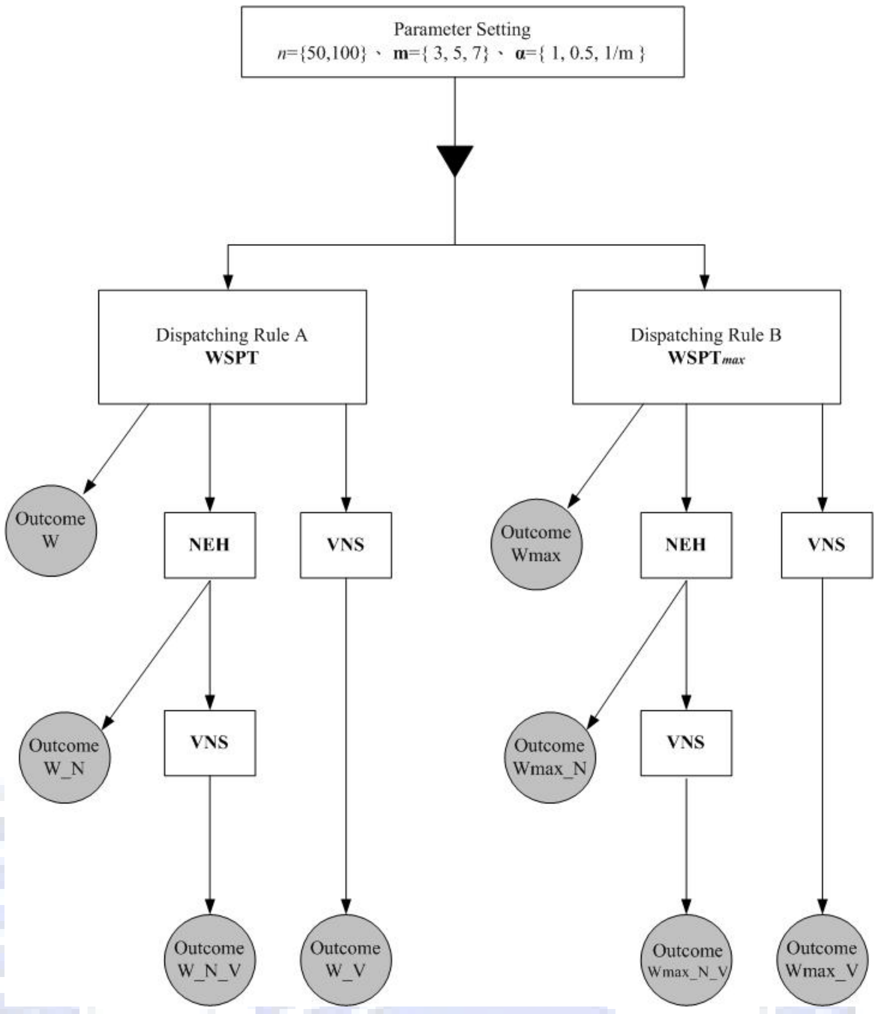


Figure 6: Experiment settings.

One instance is solved by first two dispatching rules respectively, **WSPT** and **WSPT**$_{\max}$, and we get the first two outcomes, W and $W_{\max}$. Second, We use solution W and solution $W_{\max}$ as the initial sequences to invoke the NEH-based algorithm and the VNS algorithm, respectively. Another four outcomes, W_N, W_V, $W_{\max}\_N$, and $W_{\max}\_V$ follow. Finally, solution W_N and solution $W_{\max}\_N$ are used as the initial sequences for the VNS algorithm. The solutions are denoted by W_N_V and $W_{\max}\_N\_V$. We analyze all the outcomes obtained from this experiment setting in the next section.

4.2 Computational results and analysis

The computational results shown in Tables 5 and 6 are the average values over the 10 instances for each combination of the number of orders (n), the number of machines (m) and the value of α. **Table 5** summarizes the results concerning solution quality and **Table 6** shows the running times of the proposed algorithms. Note that the running times required by the two dispatching rules are negligible and not shown in the tables. In the following, we plot figures of numerical values extracted from the tables along different aspects of observations.

4.2.1 Solution Quality

Figure 7 shows the performances of the heuristics in each setting. In all scenarios, the NEH-based algorithm can improve the solutions produced by the two dispatching rules, yet the VNS algorithm produces a relatively smaller objective value than those produced by the NEH-based approach. The VNS algorithm consistently provides better solutions than the NEH-based algorithm, no matter what initial solution was used. There are almost no differences between the objective values generated by VNS and those by NEH+VNS, even the initial solutions are different. From those figures, we can find that the performance of the dispatching rules highly depends on the α value.

To explain the phenomenon better, **Figures 8** illustrates the relationship between dispatching rules and α values with different combinations of m and n. The $\mathbf{WSPT}_{\max}$ rule produces better schedules when $\alpha = 1$, while the **WSPT** rule performs better when $\alpha = 1/m$. Furthermore, no evident difference exists between the solution quality yielded by these two rules when $\alpha = 0.5$. This phenomenon makes sense. If the weight of vendee is relatively higher than the manufacturers, planning a schedule according to the vendee's preference could have a better system-wide solution. On the contrary, if the weight of the vendee is nearly the same with the weight of anyone of the manufacturers, planning a production schedule abiding by all the manufacturers' preference would probably gain a better system-wide solution. Especially, when the number of machines increases, the difference between the outcomes from these two rules becomes more clear. Note that the data size, which includes the number of machines and the number of orders, does not have significant influence on the performances of the two dispatching rules.

The role and performance of the NEH-based heuristic are of special interests. Initial solutions and the α values appear to exhibit a certain kind of correlation. But the effect of α values on the NEH-based algorithm is not as strong as on the dispatching rules. Performing the NEH-based algorithm with an initial solution generated by the **WSPT** rule gains a better outcome if $\alpha = 1$; whereas, the $\mathbf{WSPT}_{\max}$ rule would obtain a better initial solution if $\alpha = 1/m$. Similarly, there is no significant difference between these objective values of NEH-based solutions with different initial solutions when $\alpha = 0.5$. The results are demonstrated in **Figure 9**. Different initial solutions and α values do not seem to significantly influence the performance of VNS. Specifically, the VNS algorithm with different initial solutions produce objective values which are nearly equal to one another for all values of α. The observations are revealed from the chart shown in **Figure 10**. Consequently, there is no difference between the outcomes with the two dispatching rules even both NEH and VNS are invoked, as shown in **Figure**

11.

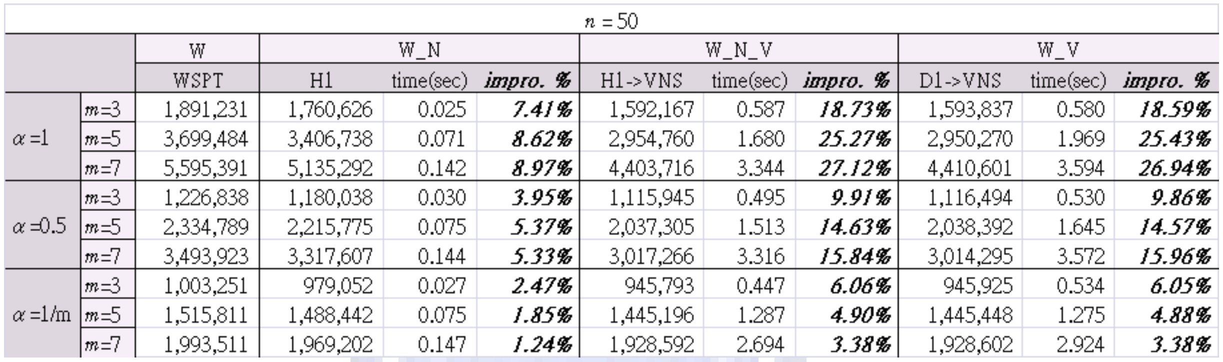
4.2.2 Analysis of run time performance

From **Table 6**, we found that the running time is not affected by the α values and the initial solutions. Accordingly, we inspect the running time performance of the heuristics from the aspect of data size only. The running time results shown in **Table 6** are the average values over the three α values and two dispatching rules for each combination of m and n. Note that the running time of a specific instance may vary from time to time due to the intrinsic randomness of the VNS algorithm.

In general, as observed from **Figure 12**, data size has a slight effect on the running time of the NEH-based algorithm, but it substantially affects the running time of the VNS-based algorithm. All of the test instances could be solved in a few seconds by the NEH-based algorithm (e.g., the average running time is 1.432 seconds when $n = 100$ and $m = 7$). The running time of VNS is positively related to the data size, such as that shown in **Figure 12**. The running time is 0.603 seconds when $n = 50$ and $m = 3$, while it takes 57.489 seconds on average to solve the instances with $n = 100$ and $m = 7$. The running time of the hybrid NEH+VNS algorithm is almost equal to that the VNS-based algorithm because the running time of the NEH-based algorithm is relatively small in comparison with that required by the VNS-based algorithm.

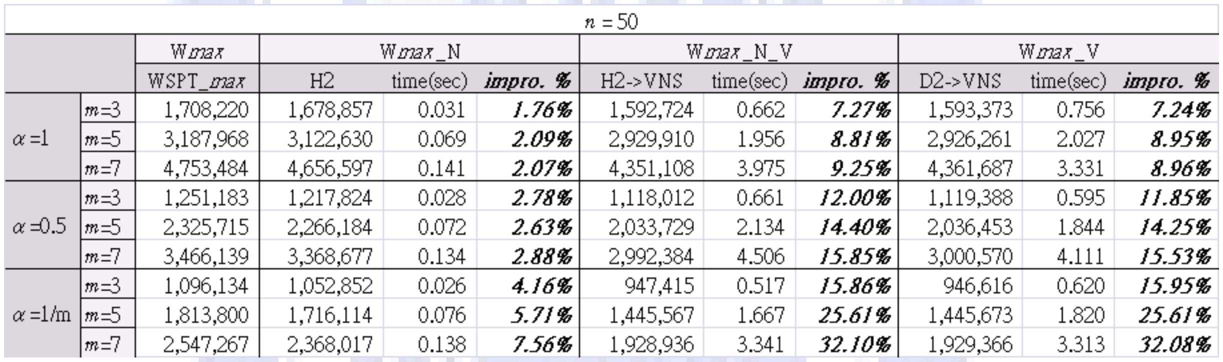
From the above analysis on solution quality and running time, we come to the conclusion that if we want to reach a scheduling plan in a short time, the **WSPT** and **WSPT**$_{\max}$ rules can be considered. For better quality, the NEH-based algorithm can be deployed using the initial solutions given by the two dispatching rules. If more decision time is allowed, the VNS approach is recommended for relatively good schedules.

Table 5: Solution values and improvement ratios.



$n = 50$											
		W	W_N			W_N_V			W_V		
		WSPT	H1	time(sec)	***impro. %***	H1->VNS	time(sec)	***impro. %***	D1->VNS	time(sec)	***impro. %***
α =1	m=3	1,891,231	1,760,626	0.025	***7.41%***	1,592,167	0.587	***18.73%***	1,593,837	0.580	***18.59%***
	m=5	3,699,484	3,406,738	0.071	***8.62%***	2,954,760	1.680	***25.27%***	2,950,270	1.969	***25.43%***
	m=7	5,595,391	5,135,292	0.142	***8.97%***	4,403,716	3.344	***27.12%***	4,410,601	3.594	***26.94%***
α =0.5	m=3	1,226,838	1,180,038	0.030	***3.95%***	1,115,945	0.495	***9.91%***	1,116,494	0.530	***9.86%***
	m=5	2,334,789	2,215,775	0.075	***5.37%***	2,037,305	1.513	***14.63%***	2,038,392	1.645	***14.57%***
	m=7	3,493,923	3,317,607	0.144	***5.33%***	3,017,266	3.316	***15.84%***	3,014,295	3.572	***15.96%***
α =1/m	m=3	1,003,251	979,052	0.027	***2.47%***	945,793	0.447	***6.06%***	945,925	0.534	***6.05%***
	m=5	1,515,811	1,488,442	0.075	***1.85%***	1,445,196	1.287	***4.90%***	1,445,448	1.275	***4.88%***
	m=7	1,993,511	1,969,202	0.147	***1.24%***	1,928,592	2.694	***3.38%***	1,928,602	2.924	***3.38%***

(a) $n = 50$, initial solutions generated by WSPT.



$n = 50$											
		W*max*	W*max*_N			W*max*_N_V			W*max*_V		
		WSPT_*max*	H2	time(sec)	***impro. %***	H2->VNS	time(sec)	***impro. %***	D2->VNS	time(sec)	***impro. %***
α =1	m=3	1,708,220	1,678,857	0.031	***1.76%***	1,592,724	0.662	***7.27%***	1,593,373	0.756	***7.24%***
	m=5	3,187,968	3,122,630	0.069	***2.09%***	2,929,910	1.956	***8.81%***	2,926,261	2.027	***8.95%***
	m=7	4,753,484	4,656,597	0.141	***2.07%***	4,351,108	3.975	***9.25%***	4,361,687	3.331	***8.96%***
α =0.5	m=3	1,251,183	1,217,824	0.028	***2.78%***	1,118,012	0.661	***12.00%***	1,119,388	0.595	***11.85%***
	m=5	2,325,715	2,266,184	0.072	***2.63%***	2,033,729	2.134	***14.40%***	2,036,453	1.844	***14.25%***
	m=7	3,466,139	3,368,677	0.134	***2.88%***	2,992,384	4.506	***15.85%***	3,000,570	4.111	***15.53%***
α =1/m	m=3	1,096,134	1,052,852	0.026	***4.16%***	947,415	0.517	***15.86%***	946,616	0.620	***15.95%***
	m=5	1,813,800	1,716,114	0.076	***5.71%***	1,445,567	1.667	***25.61%***	1,445,673	1.820	***25.61%***
	m=7	2,547,267	2,368,017	0.138	***7.56%***	1,928,936	3.341	***32.10%***	1,929,366	3.313	***32.08%***

(b) $n = 50$, initial solutions generated by $\mathrm{WSPT}_{\max}$.

$n = 100$												
		W	W_N			W_N_V			W_V			
		WSPT	H1	time(sec)	***impro. %***	H1->VNS	time(sec)	***impro. %***	D1->VNS	time(sec)	***impro. %***
α =1	m=3	7,397,369	6,899,812	0.302	***7.19%***	6,153,096	8.797	***20.18%***	6,161,674	8.080	***20.02%***
	m=5	14,297,404	13,181,720	0.783	***8.45%***	11,189,648	25.763	***27.75%***	11,200,787	28.158	***27.63%***
	m=7	21,307,079	19,693,146	1.437	***8.20%***	16,567,155	49.838	***28.61%***	16,527,062	54.736	***28.93%***
α =0.5	m=3	4,753,425	4,573,514	0.300	***3.93%***	4,295,657	6.677	***10.64%***	4,292,676	7.995	***10.71%***
	m=5	8,979,190	8,576,168	0.786	***4.69%***	7,773,596	22.364	***15.50%***	7,768,581	24.873	***15.56%***
	m=7	13,269,426	12,637,781	1.439	***5.01%***	11,351,786	54.084	***16.89%***	11,361,411	53.931	***16.80%***
α =1/m	m=3	3,874,491	3,784,079	0.304	***2.39%***	3,638,257	6.338	***6.48%***	3,639,348	6.361	***6.45%***
	m=5	5,788,746	5,693,867	0.787	***1.66%***	5,511,716	20.027	***5.02%***	5,511,783	20.395	***5.02%***
	m=7	7,535,270	7,448,634	1.434	***1.16%***	7,265,389	37.159	***3.71%***	7,266,624	34.797	***3.70%***

(c) $n = 100$, initial solutions generated by WSPT.

$n = 100$												
		W*max*	W*max*_N			W*max*_N_V			W*max*_V			
		WSPT_*max*	H2	time(sec)	***impro. %***	H2->VNS	time(sec)	***impro. %***	D2->VNS	time(sec)	***impro. %***
α =1	m=3	6,681,234	6,585,749	0.292	***1.44%***	6,158,900	10.670	***8.49%***	6,159,114	10.648	***8.49%***
	m=5	12,235,240	12,044,150	0.744	***1.59%***	11,138,553	40.145	***9.90%***	11,141,943	33.358	***9.86%***
	m=7	17,882,205	17,635,571	1.425	***1.39%***	16,323,224	69.194	***9.55%***	16,314,098	67.956	***9.61%***
α =0.5	m=3	4,897,862	4,788,201	0.289	***2.29%***	4,300,631	9.389	***13.90%***	4,301,013	8.967	***13.89%***
	m=5	8,968,068	8,790,201	0.736	***2.04%***	7,760,570	38.353	***15.61%***	7,758,385	38.681	***15.65%***
	m=7	13,130,781	12,880,818	1.420	***1.94%***	11,274,898	81.908	***16.45%***	11,280,621	76.411	***16.39%***
α =1/m	m=3	4,310,062	4,163,851	0.296	***3.52%***	3,643,479	7.891	***18.32%***	3,644,073	7.572	***18.30%***
	m=5	7,028,507	6,691,854	0.755	***5.06%***	5,515,087	28.802	***27.52%***	5,514,717	28.987	***27.53%***
	m=7	9,771,601	9,243,670	1.437	***5.71%***	7,275,832	55.556	***34.30%***	7,262,765	57.102	***34.54%***

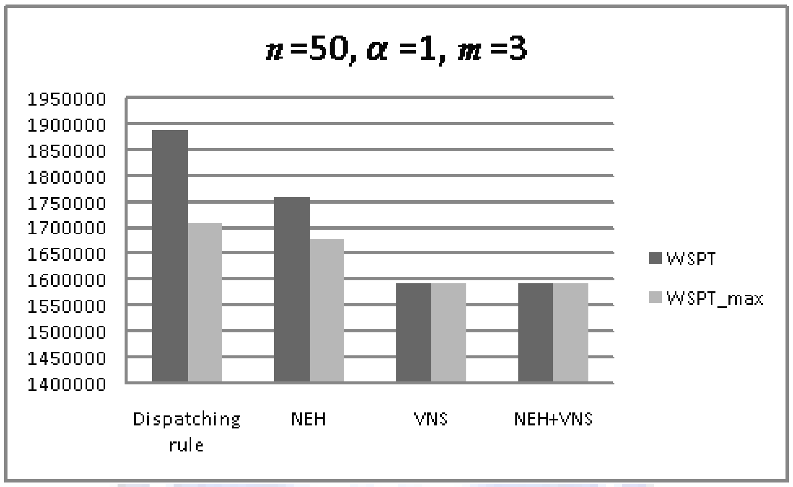
(d) $n = 100$, initial solutions generated by $\text{WSPT}_{\max}$.

Table 6: Elapsed running time.

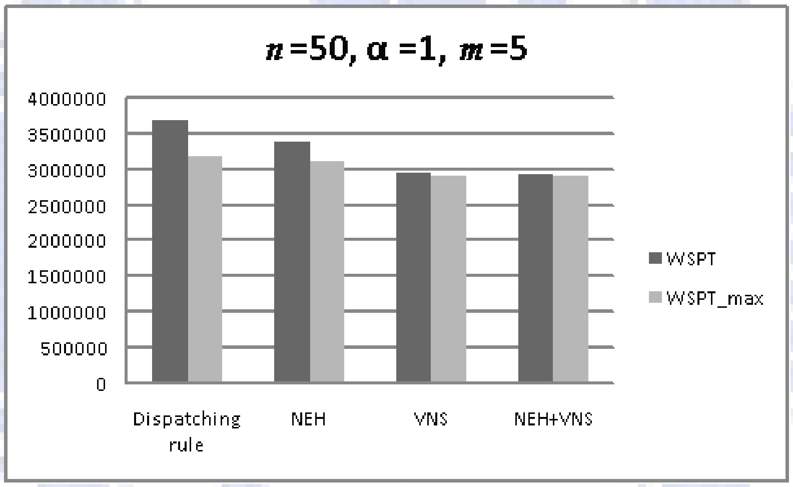
(unit: sec)	n =50			n =100		
	NEH	NEH+VNS	VNS	NEH	NEH+VNS	VNS
m =3	0.02770	0.56162	0.60255	0.29710	8.29348	8.27065
m =5	0.07295	1.70620	1.76328	0.76502	29.24228	29.07547
m =7	0.14097	3.52908	3.47398	1.43218	57.95657	57.48878

Figure 7: Performance of heuristics in different settings.

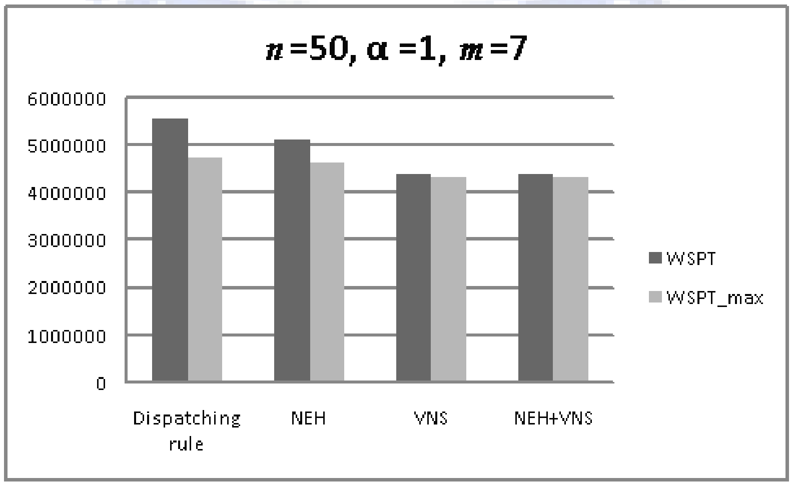
(a)



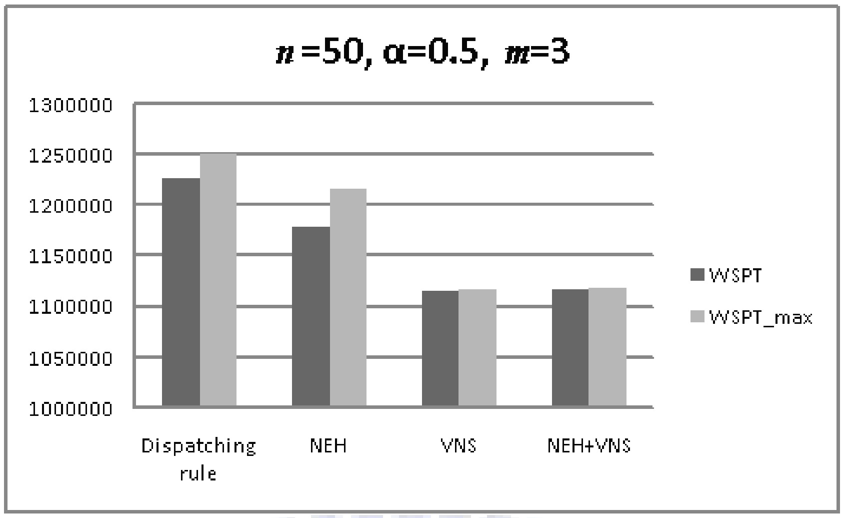
(b)



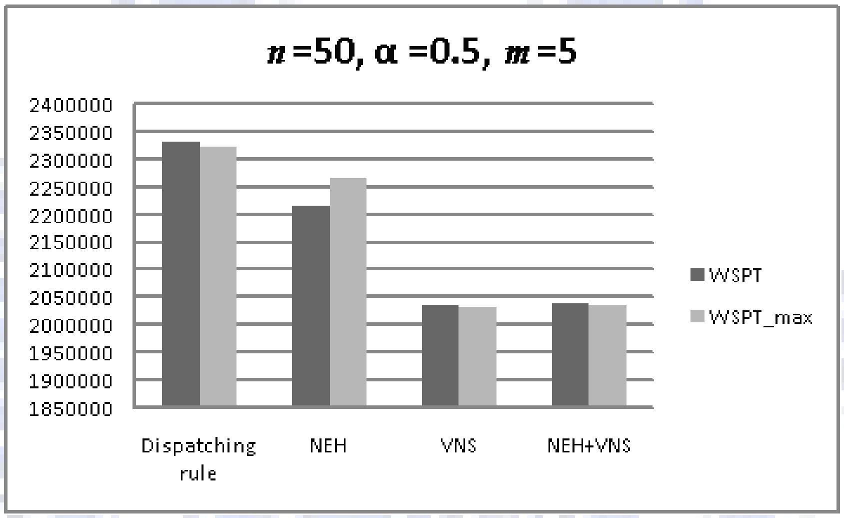
(c)



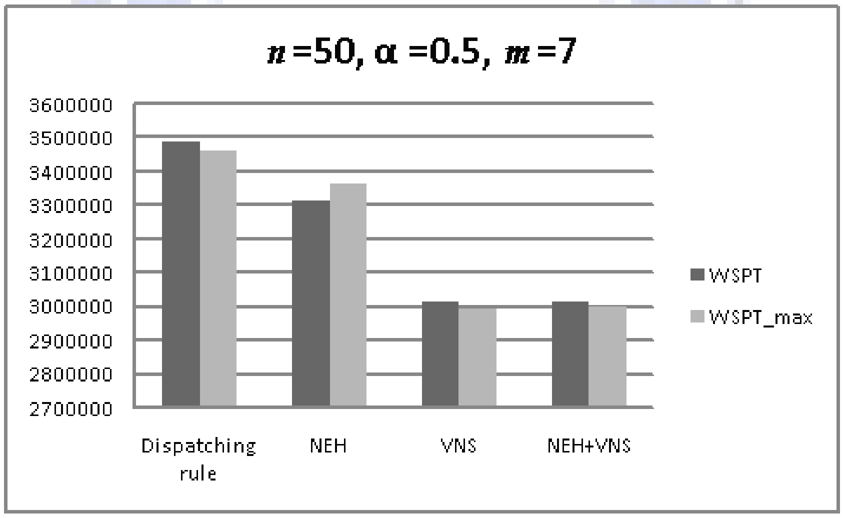
(d)


n =50, α=0.5, m=3
1300000
1250000
1200000
1150000
1100000
1050000
1000000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

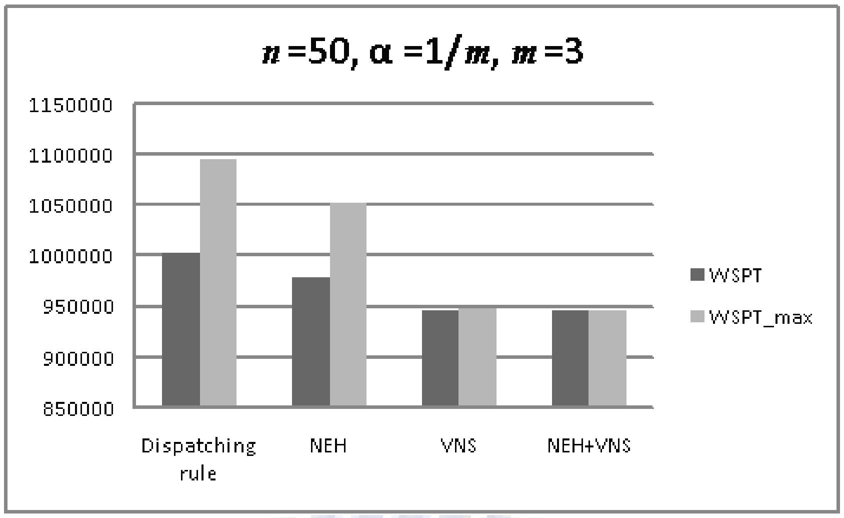
(e)


n =50, α =0.5, m =5
2400000
2350000
2300000
2250000
2200000
2150000
2100000
2050000
2000000
1950000
1900000
1850000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(f)

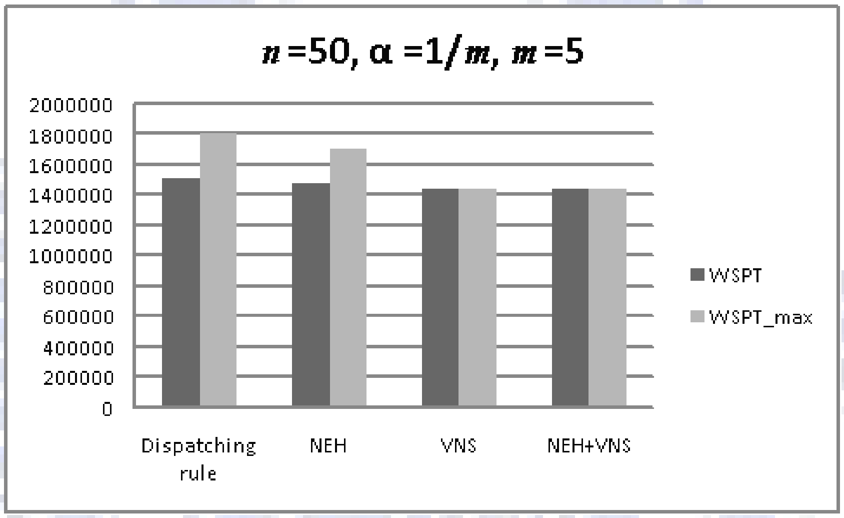

n =50, α =0.5, m =7
3600000
3500000
3400000
3300000
3200000
3100000
3000000
2900000
2800000
2700000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(g)



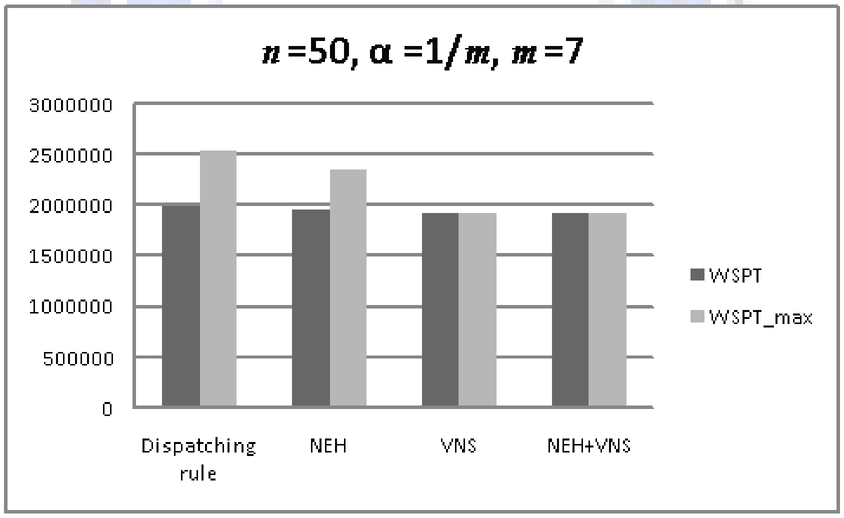
n =50, α =1/*m*, *m* =3
1150000
1100000
1050000
1000000
950000
900000
850000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(h)



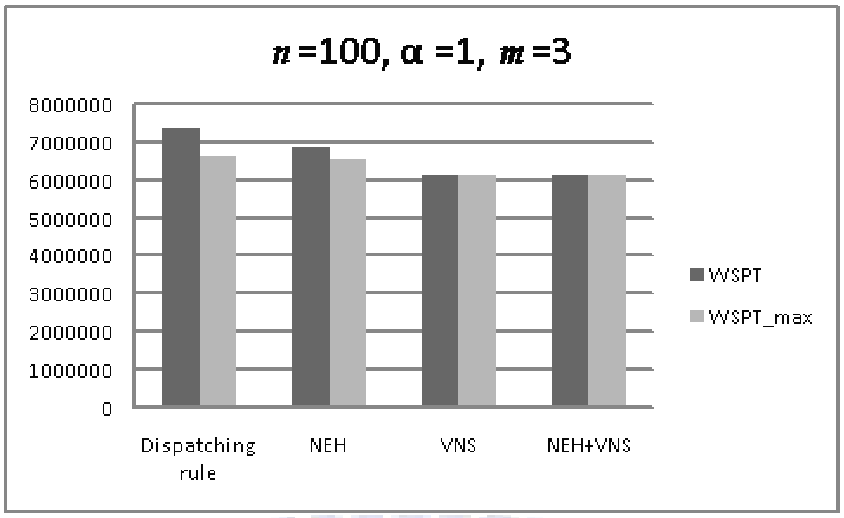
n =50, α =1/*m*, *m* =5
2000000
1800000
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(i)

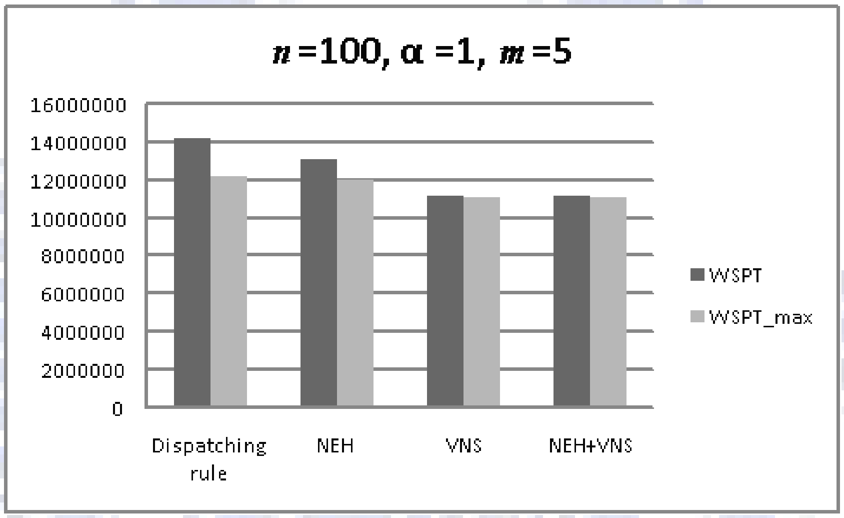


n =50, α =1/*m*, *m* =7
3000000
2500000
2000000
1500000
1000000
500000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

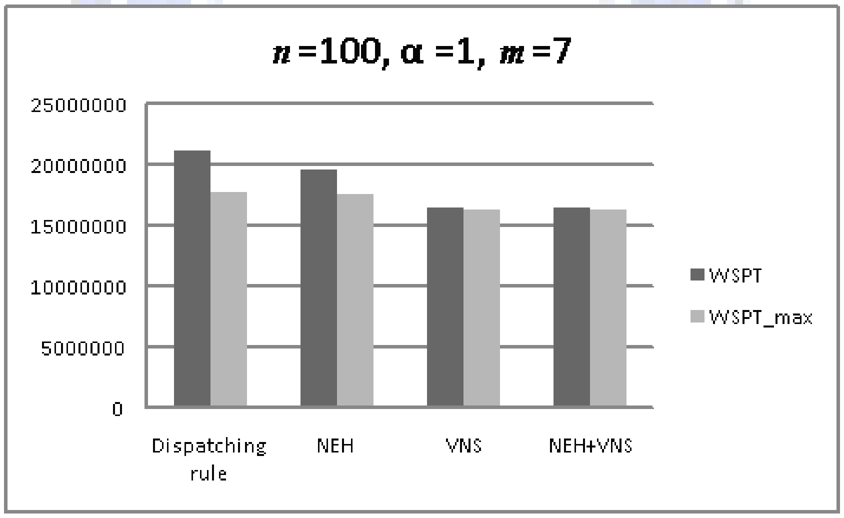
(j)


n =100, α =1, *m* =3
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

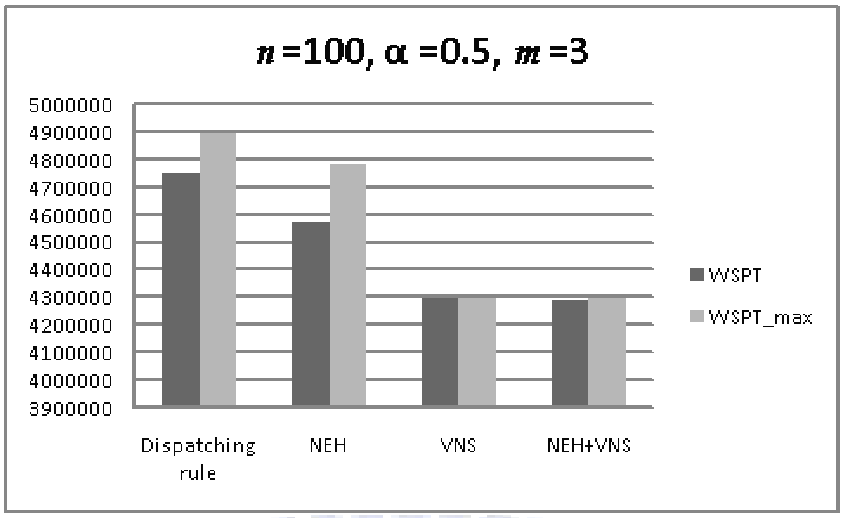
(k)


n =100, α =1, *m* =5
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

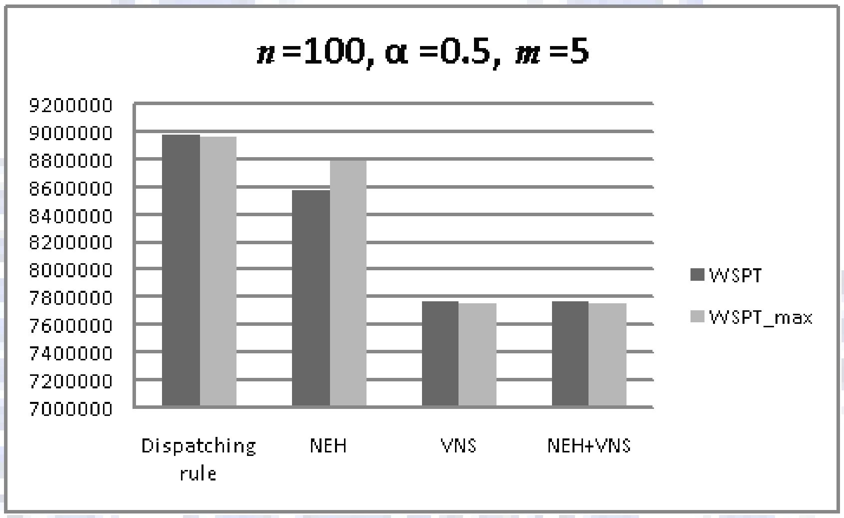
(l)


n =100, α =1, *m* =7
25000000
20000000
15000000
10000000
5000000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

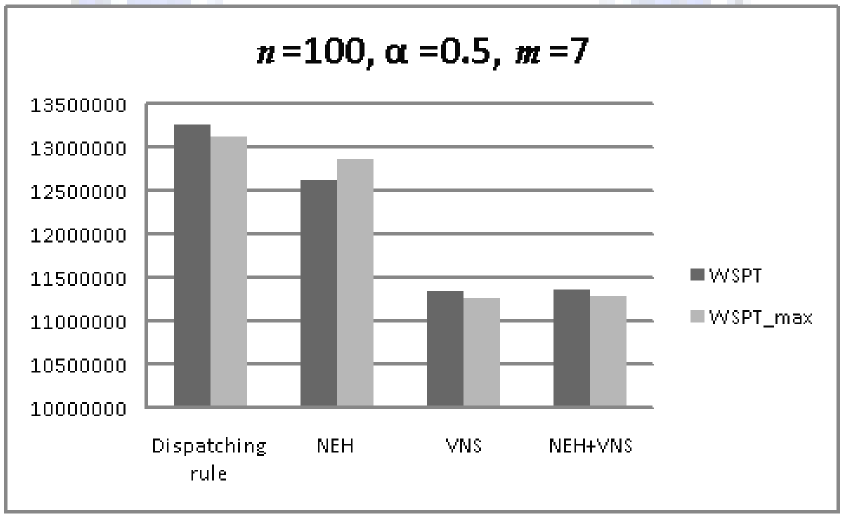
(m)


n =100, α =0.5, m =3
5000000
4900000
4800000
4700000
4600000
4500000
4400000
4300000
4200000
4100000
4000000
3900000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

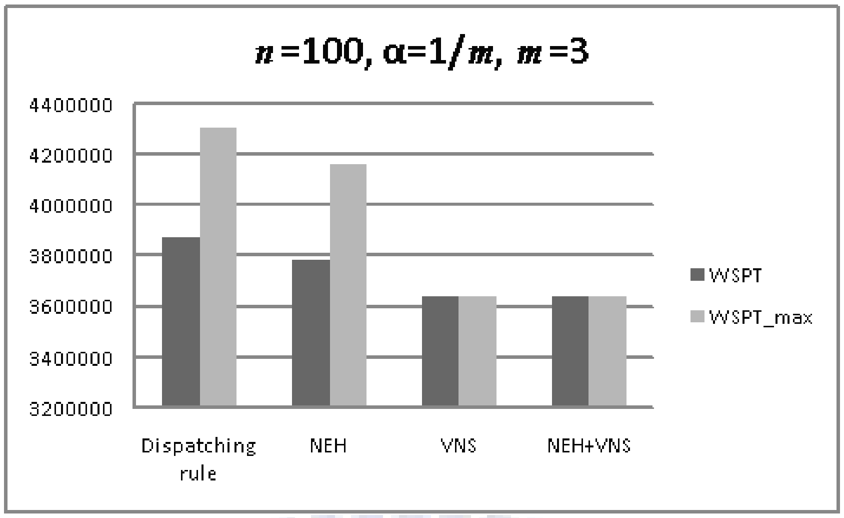
(n)


n =100, α =0.5, m =5
9200000
9000000
8800000
8600000
8400000
8200000
8000000
7800000
7600000
7400000
7200000
7000000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

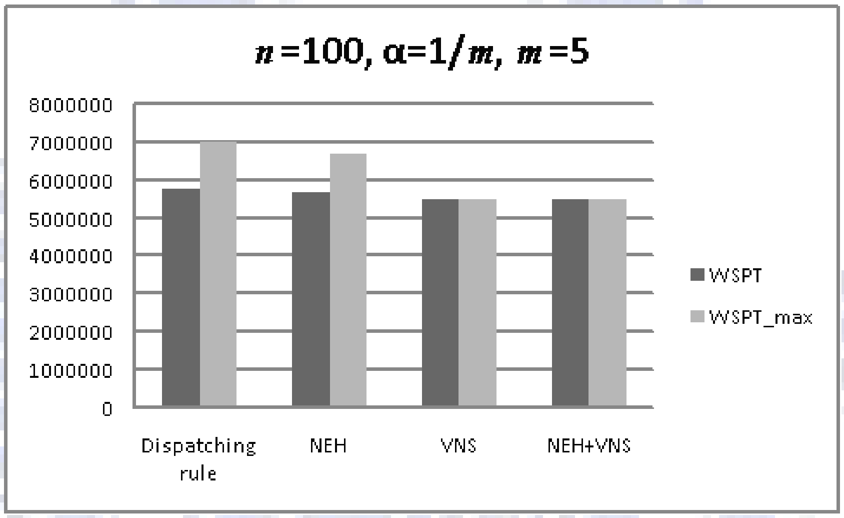
(o)


n =100, α =0.5, m =7
13500000
13000000
12500000
12000000
11500000
11000000
10500000
10000000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(p)


n=100, α=1/m, m=3
4400000
4200000
4000000
3800000
3600000
3400000
3200000
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(q)


n=100, α=1/m, m=5
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

(r)

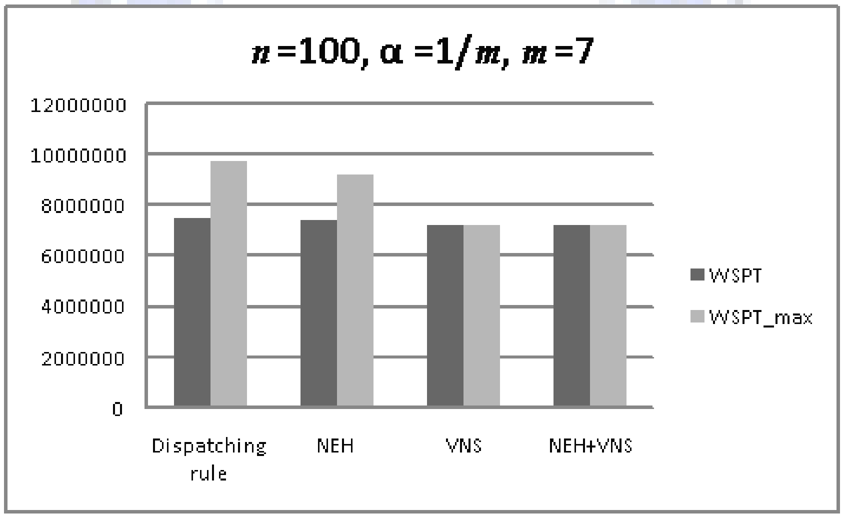
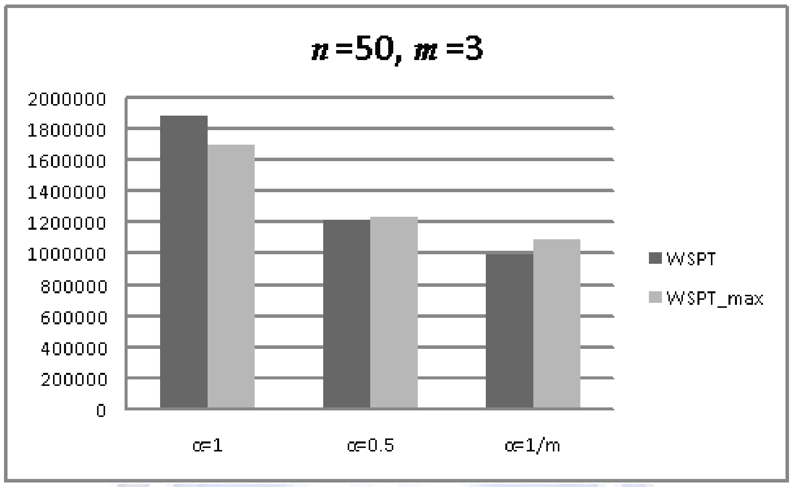
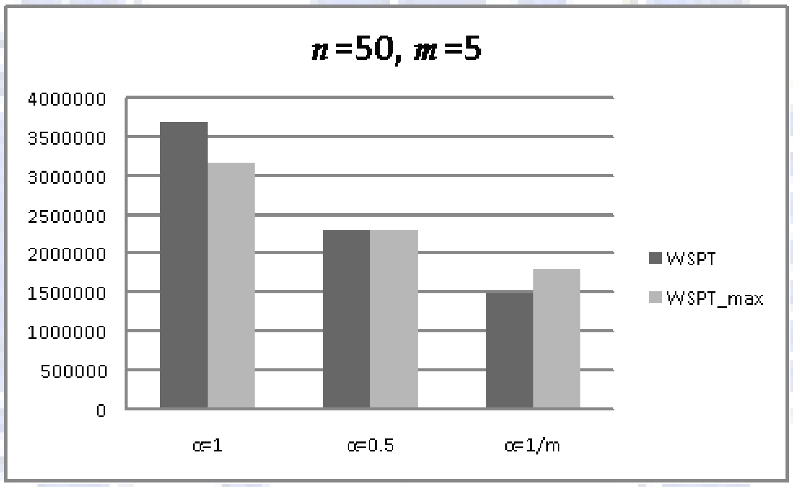

n=100, α =1/m, m=7
12000000
10000000
8000000
6000000
4000000
2000000
0
Dispatching rule
NEH
VNS
NEH+VNS
WSPT
WSPT_max

Figure 8: Correlations between α value and performance of dispatching rules.

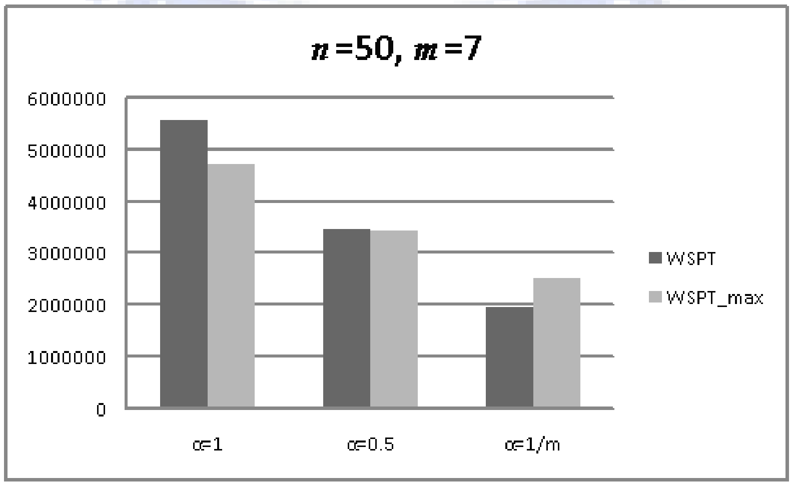
(a)



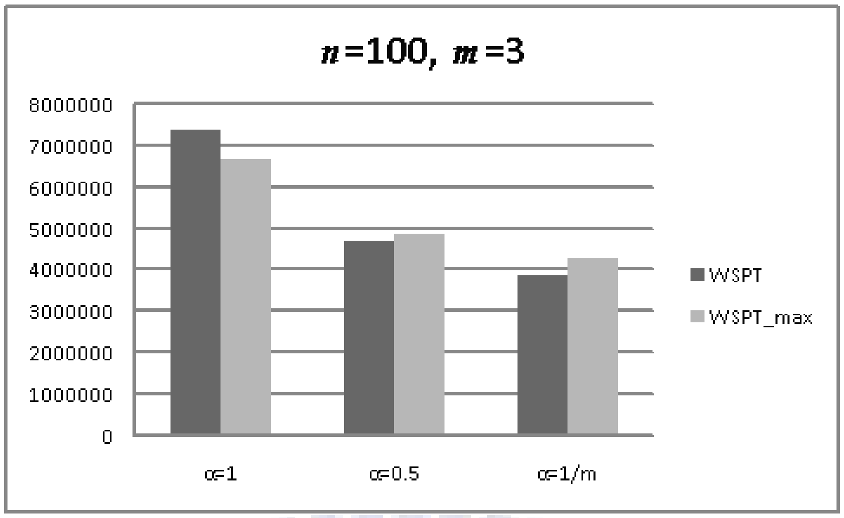
(b)



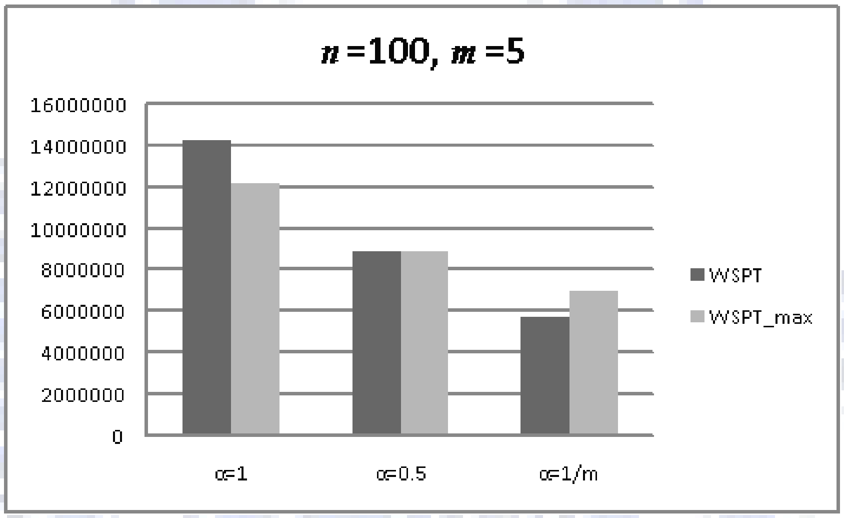
(c)



(d)


n =100, *m* =3
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

(e)


n =100, *m* =5
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

(f)

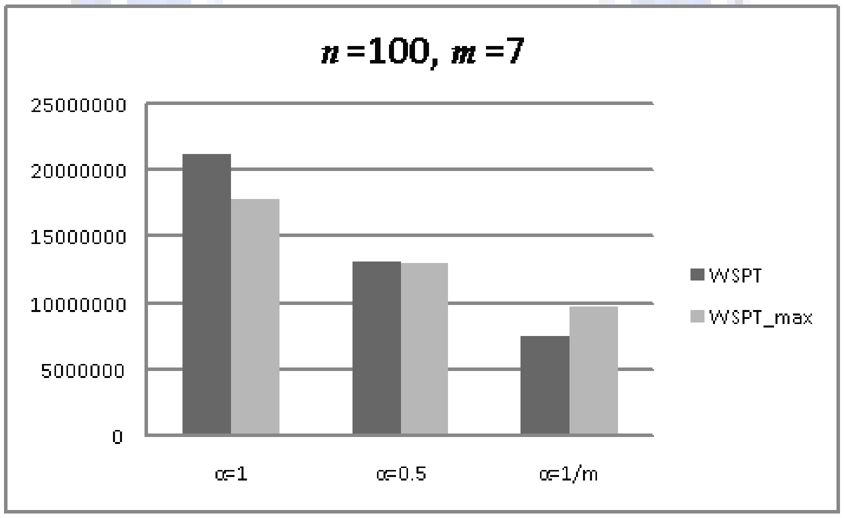
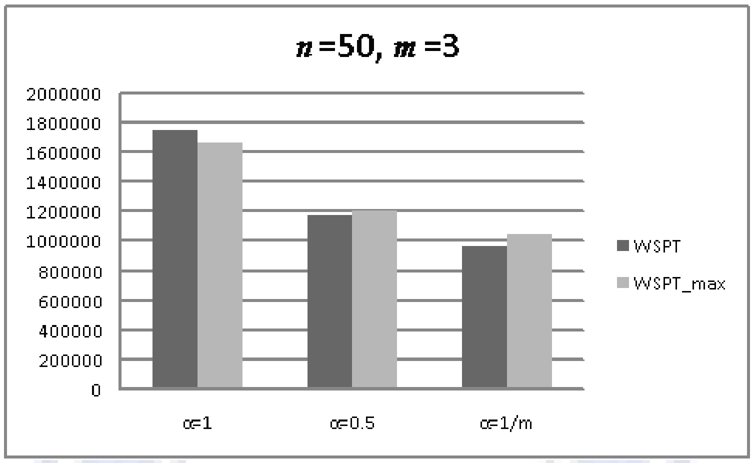
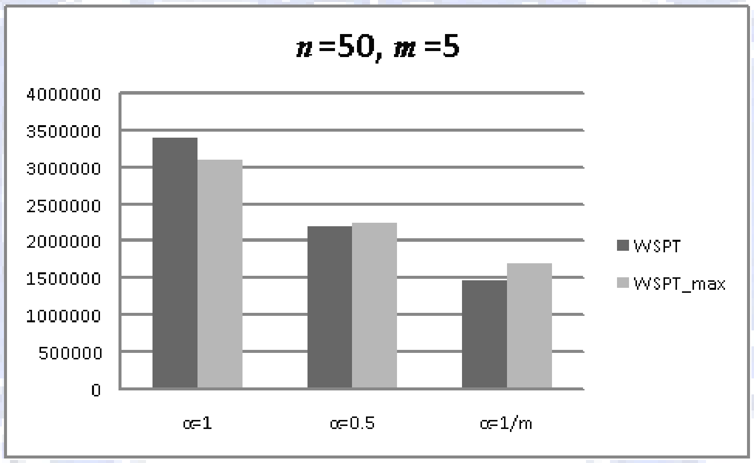

n =100, *m* =7
25000000
20000000
15000000
10000000
5000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

Figure 9: Correlations between α and performance of NEH with different initial solutions.

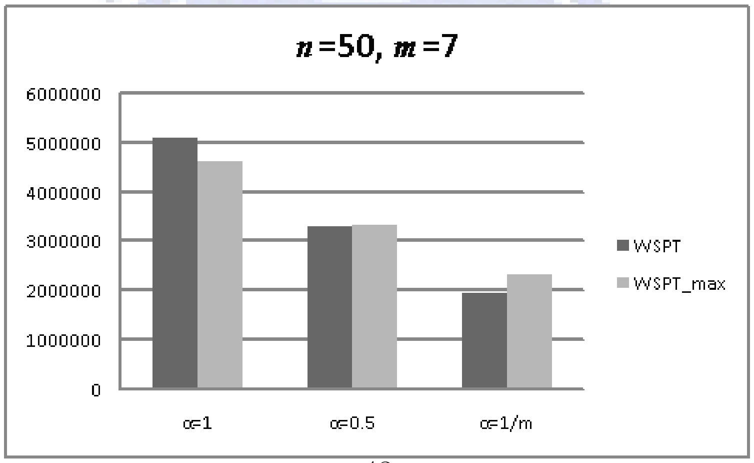
(a)



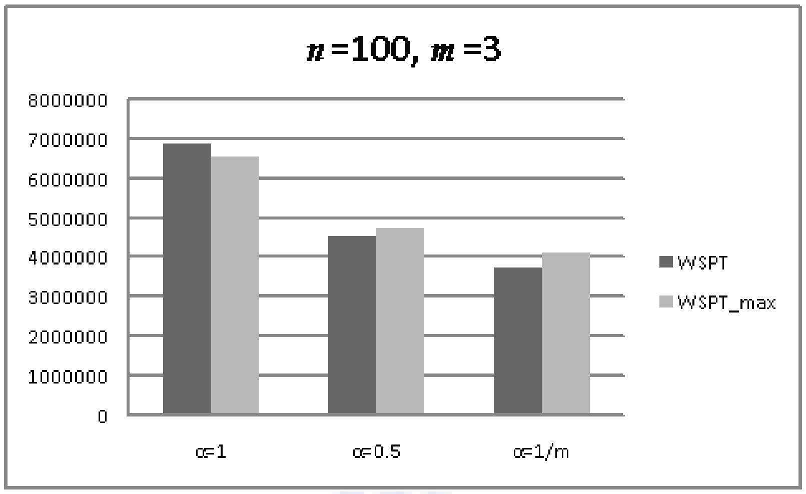
(b)



(c)

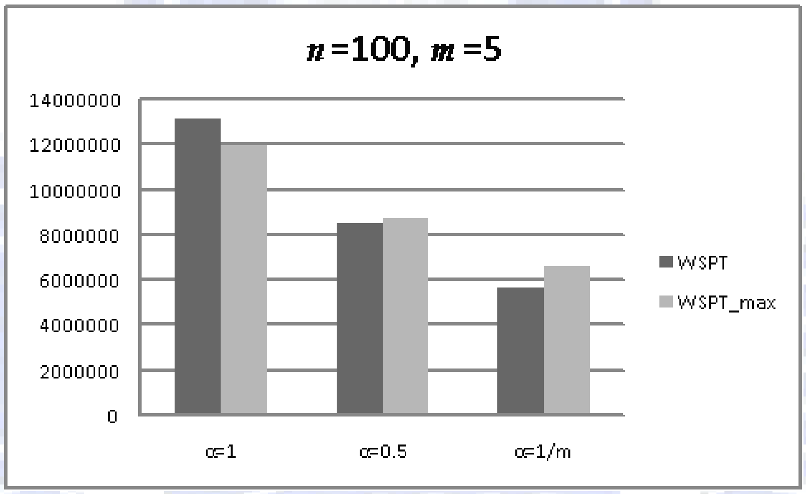


(d)



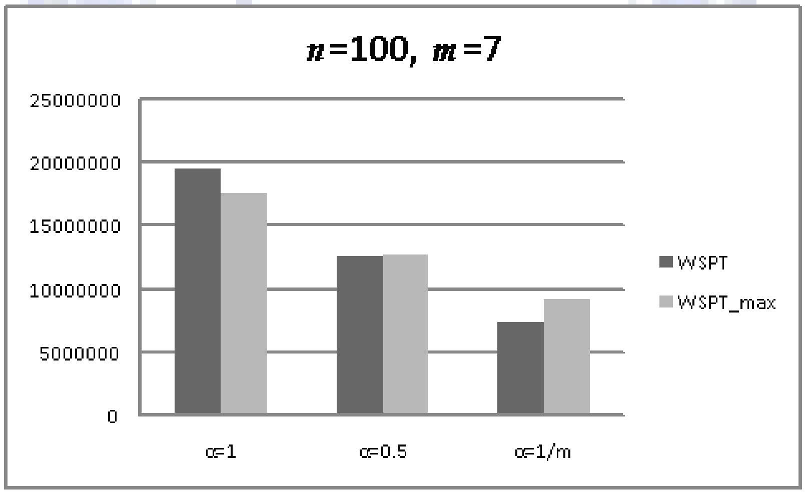
n=100, m=3
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

(e)



n=100, m=5
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

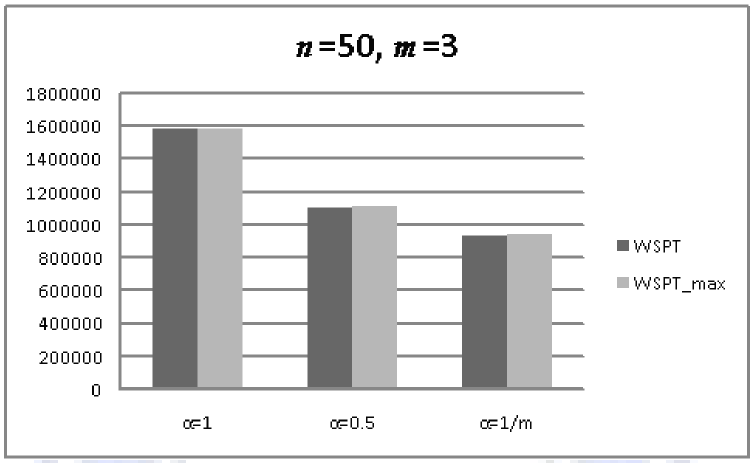
(f)



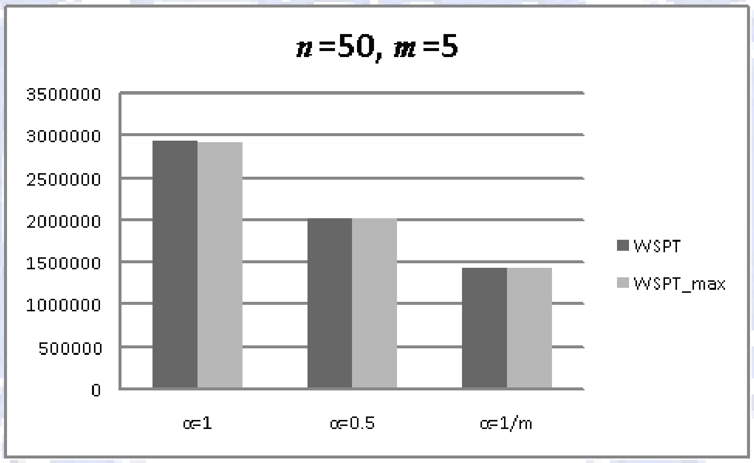
n=100, m=7
25000000
20000000
15000000
10000000
5000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

Figure 10: Correlations between α value and performance of NEH+VNS with different initial solutions.

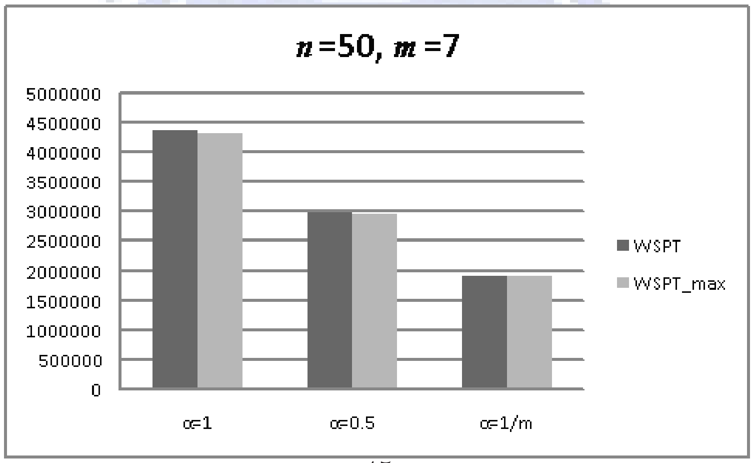
(a)



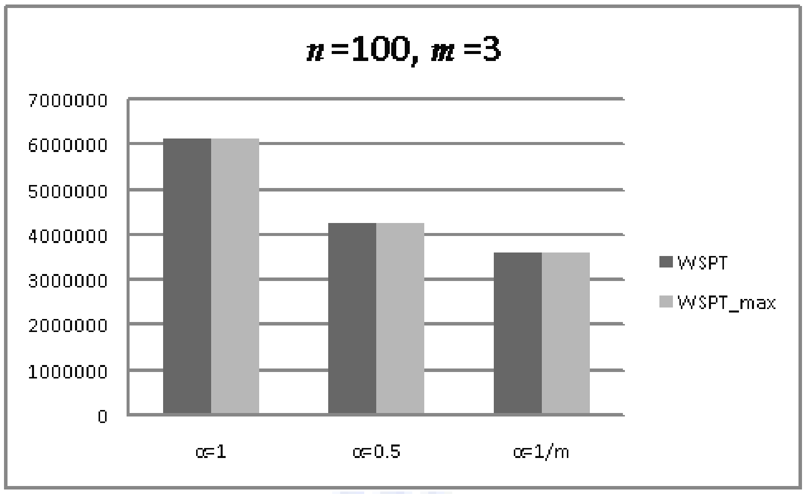
(b)



(c)

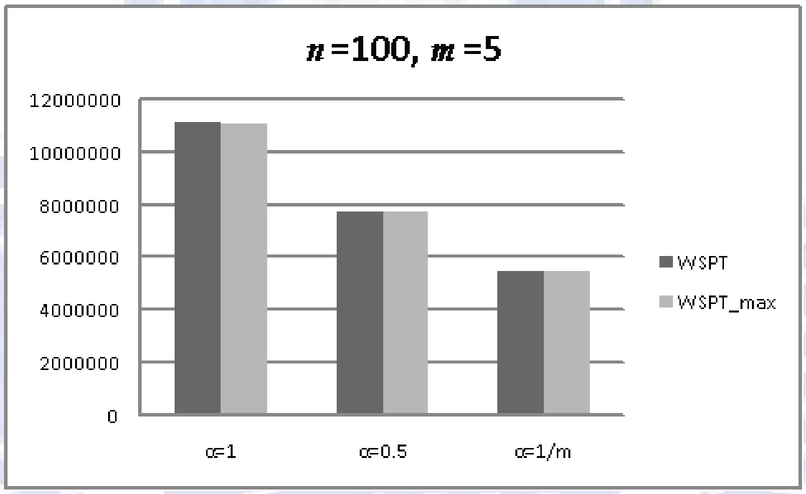


(d)



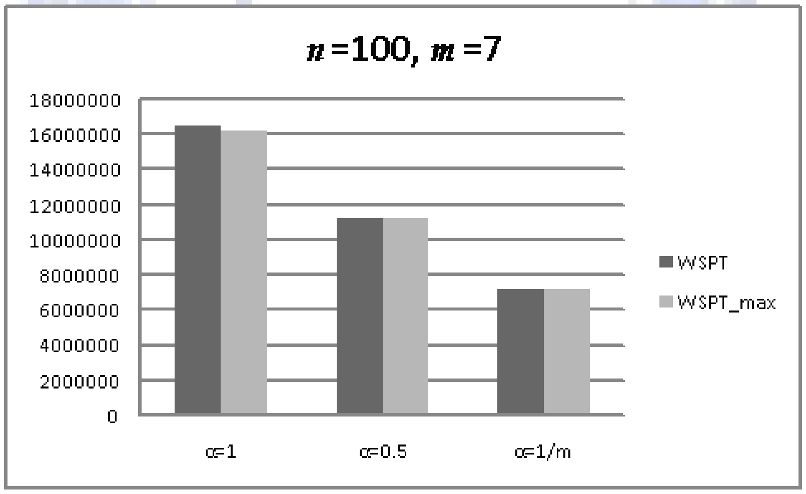
n =100, m =3
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

(e)



n =100, m =5
12000000
10000000
8000000
6000000
4000000
2000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

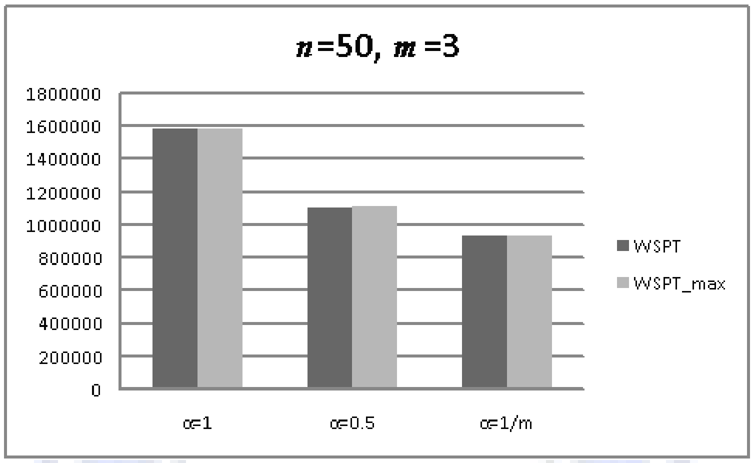
(f)



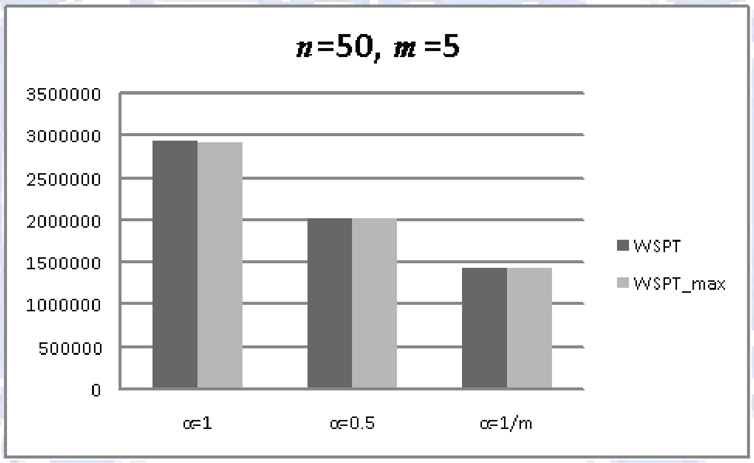
n =100, m =7
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

Figure 11: Correlations between α and performance of VNS with different initial solutions.

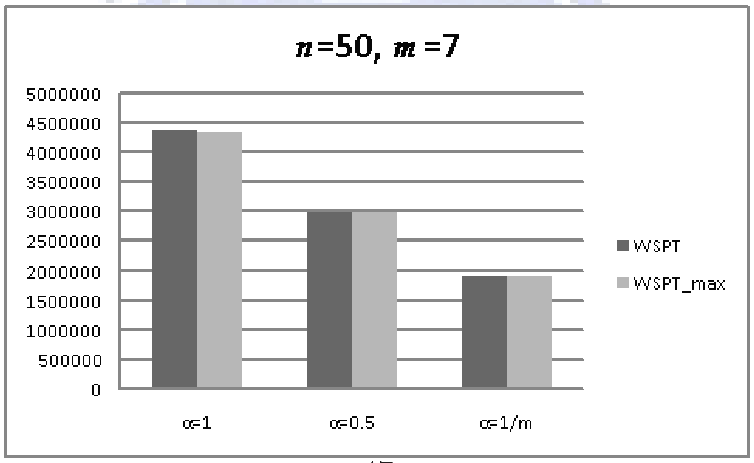
(a)



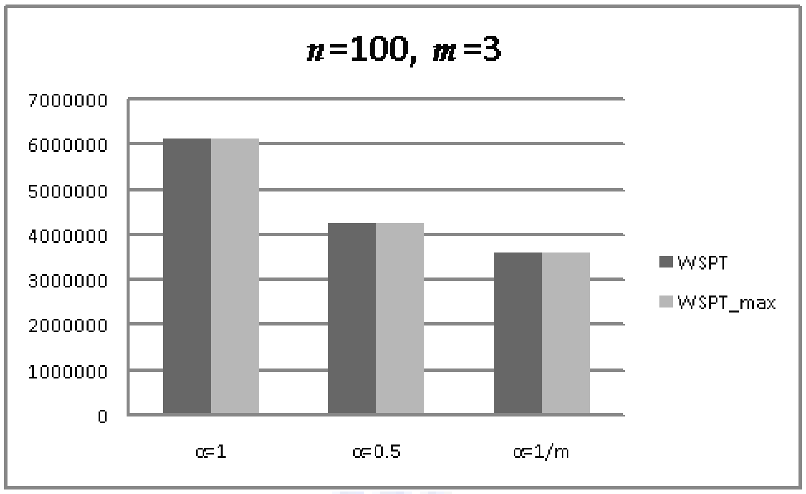
(b)



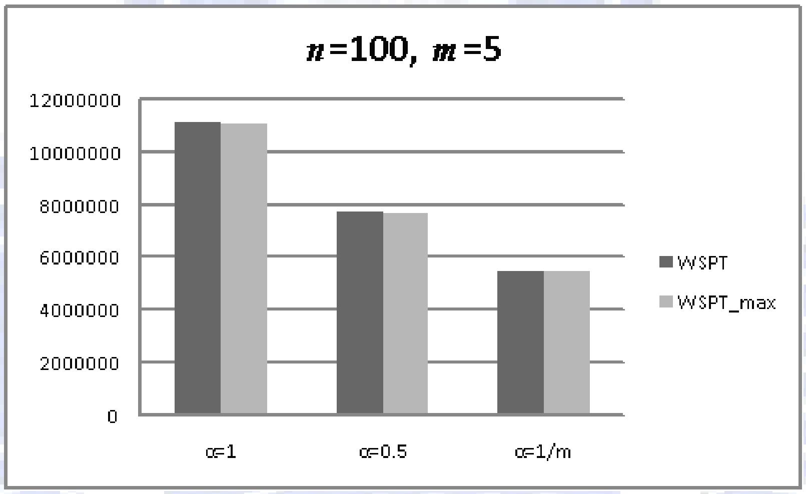
(c)



(d)


n=100, *m*=3
7000000
6000000
5000000
4000000
3000000
2000000
1000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

(e)


n=100, *m*=5
12000000
10000000
8000000
6000000
4000000
2000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

(f)

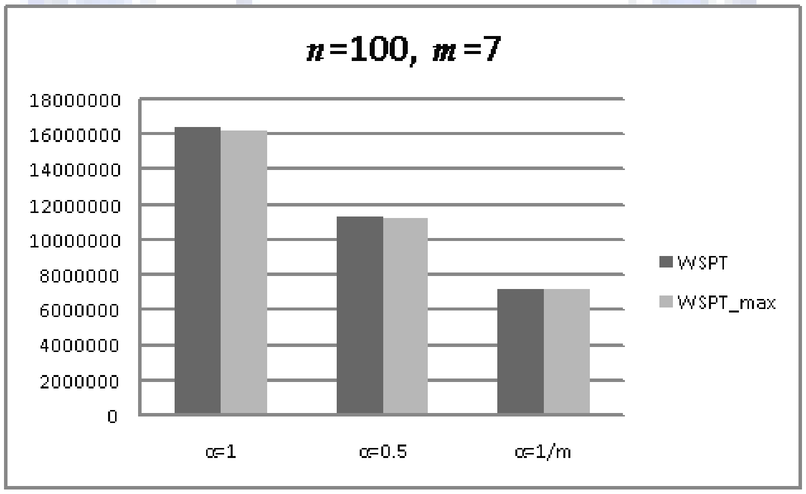
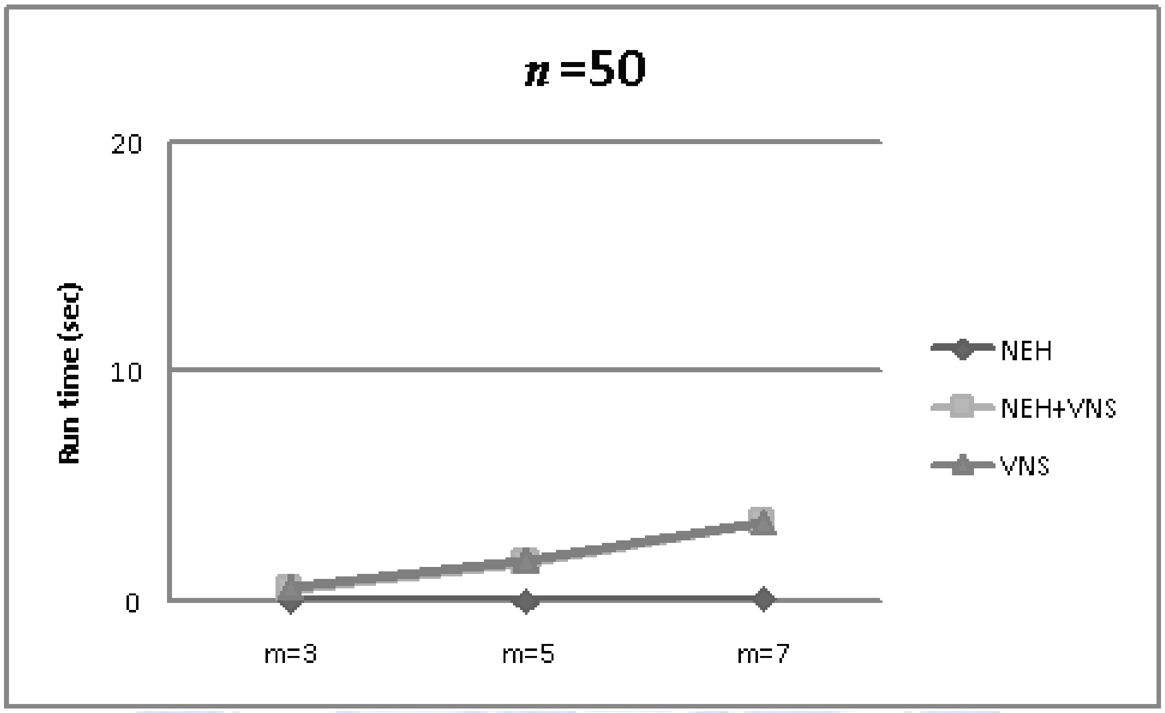
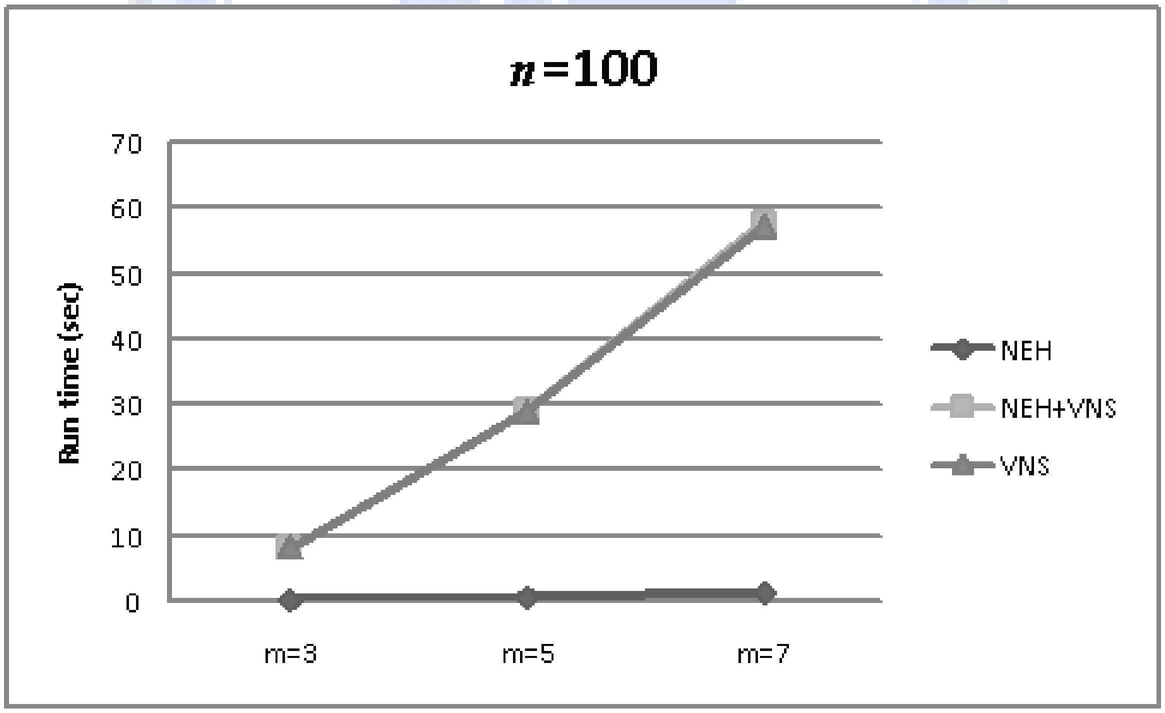

n=100, *m*=7
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
α=1
α=0.5
α=1/m
WSPT
WSPT_max

Figure 12: Correlations between data size and running time

(a)



(b)



Chapter 5

Concluding Remarks

In this thesis, we discussed a scheduling problem in concurrent open shops incorporating coordination among all participants. Weights are assigned to not only orders (jobs) but also product items (operations) to reflect the fact that any operations, parts or items need to be included in the formation of objective functions and the fact that the positions of buyers and sellers may not be equal in real-world applications, such as buyers' market or sellers' market.

We first gave two integer programming formulations, based on positional variables and sequencing variables, of the studied problem. While the problem is computationally intractable, we proposed a three-phase algorithm to produce approximate solutions. Our algorithm comprises of a WSPT heuristic, an NEH-based algorithm and the variable neighborhood search algorithm. From computational experiments, we found that the VNS-based algorithm achieves a satisfactory performance in all test instances; however, the required running time grows significantly when the data size increases. The average running time of the largest instance, with n=100 and m=7, is around 60 seconds. It is much longer in comparison to the time of less than one second elapsed when solving the instances with small-size instances with $n = 50$ and $m = 3$.

From the view point of production model, we assumed an order is finished as long as all of its operations are finished. The problem can be extended by presuming that the vendee needs to spend processing time p_i to assemble all the operations in an order before the order can be delivered. In this case, the processing sequence on the vendee's machine should be considered as well. Such a production model is very likely to occur in the real world and worthy of further research.

From the participants' view point of interests, we only considered the conflict interests between vendee and manufacturers in the study. Solution to the conflict between vendee and manufacturers is the focal point in this thesis. Additionally, the conflict of interests may exist between manufacturers in a real-world context. For this reason, coordination becomes more complicated since the interests of individuals in the production system are included. This is an interesting setting that we can discuss further in the future.

Optimization of individuals is not necessarily practical in the real world because no one can be self-content and other factors such as partners should be taken into consideration. Therefore, to plan an optimal schedule we have to explore the potential synergy of coordination among individuals. Coming up with a win-win strategy to create maximum profits is the trend in the field of optimization. Coordinated scheduling indeed permits considerable room for further studies.

Bibliography

[1] R.H. Ahmadi and U. Bagchi (1990). Scheduling of multi-job customer orders in multi-machine environments, *ORSA/TIMS*, Philadelphia.

[2] I. Averbakh (2009). On-line integrated productionVdistribution scheduling problems with capacitated deliveries, *European Journal of Operational Research*, doi:10.1016/j.ejor.2008.12.030.

[3] Z.-L. Chen and N.G. Hall (2007). Supply chain scheduling: Conflict and cooperation in assembly system, *Operations Research* , 55(6): 1072-1089.

[4] T.C.E. Cheng, M.Y. Kovalyov and B.M.T. Lin (1997). Single machine sheduling to minimize the batch delivery and job earliness penalty, *SIAM Journal on Optimization*, 7(2): 547-559.

[5] N.G. Hall and Z. Liu (2008). Cooperative and noncooperative games for capacity planning and scheduling, *Tutorials in Operations Research*, 108-130.

[6] P. Hansen and N. Mladenovic (2001). Variable neighborhood search: Principles and applications, *Omega*, 130(3): 449-467.

[7] C.Y. Lee, T.C.E. Cheng and B.M.T. Lin (1993). Minimizing the makespan in three-machine assembly type flow shop problem, *Management Science*, 39(5): 616–625.

[8] J.Y.T. Leung, H. Li and M. Pinedo (2005a). Order scheduling in an environment with dedicated resources in parallel, *Journal of Scheduling*, 8(5): 355–386.

[9] J.Y.T. Leung, H. Li and M. Pinedo (2006). Scheduling multiple product types with due date related objectives, *European Journal of Operational Research*, 168(2): 370–389.

[10] J.Y.T. Leung, H. Li and M. Pinedo (2007). Scheduling orders for multiple product types to minimize total weighted completion time, *Discrete Applied Mathematics*, 155(8): 945–970.

[11] J.Y.T. Leung, H. Li, M. Pinedo and S. Sriskandarajah (2005b). Open shops with jobs overlap - Revisited, *European Journal of Operational Research*, 163(2): 569–571.

[12] B.M.T. Lin and A.V. Kononov (2007). Customer order scheduling to minimize the number of late jobs, *European Journal of Perational Research* , 183(2): 944-948.

[13] U.V. Manoj, Jatinder N.D Gupta, Sushil K. Gupta (2008). Supply chain scheduling: Just-in-time environment, *Ann Oper Res*, 161: 53-86.

[14] N. Mladenovi and P. Hansen (1997). Variable neighborhood search, *Computers & Operations Research*, 24(11): 1097-1100.

[15] M. Nawaz, E.E. Emscore, Jr. and I. Ham (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega*, 11(10): 91-95.

[16] C.N. Potts and M.Y. Kovalyov (2000). Scheduling with batching: A review, *European Journal of Operational Research*, 120(2): 228–249.

[17] T.A. Roemer (2006). A note on th complexity of the concurrent open shop problem, *Journal of Scheduling* , 9(4): 389-396.

[18] T. Sawik (2008). Monolithic versus hierarchical approach to integrated scheduling in a supply chain, *International Journal of Production Research*, DOI: 10.1080/00207540802193181.

[19] G. Wang and T.C.E. Cheng (2007). Customer order scheduling to minimize total weighted completion time, *Omega* , 35(5): 623-626.