INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
A multiattribute approach to general flowshop problems

Ramazani Khorshid-Doust, Reza, Ph.D.

Case Western Reserve University, 1991
A MULTIATTRIBUTE APPROACH TO
GENERAL FLOWSHOP PROBLEMS

by

REZA RAMAZANI KHORSHID-DOUST

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Thesis Advisors:
Professor Lucien Duckstein and Professor Mihajlo D. Mesarovic

Department of Systems Engineering
CASE WESTERN RESERVE UNIVERSITY
August, 1991
CASE WESTERN RESERVE UNIVERSITY

GRADUATE STUDIES

We hereby approve the thesis of

Reza Ramezani Khorshid-Doust

candidate for the Ph.D.

degree.*

Signed: [signature]
(Chairman)

[Signature]

[Signature]

Date May 25, 1991

*We also certify that written approval has
been obtained for any proprietary material
contained therein.
A MULTIATTRIBUTE APPROACH TO
GENERAL FLOWSHOP PROBLEMS

Abstract

by

REZA RAMEZANI KHORSHID-DOUST

This dissertation is concerned with a new modeling and solution for the general flowshop problems. This class of problems covers 20 percent of the scheduling problems in industry, and is considered as a subclass of job shop problems. The problems are combinatorial in nature and NP-complete. The new modeling is applied to a special type of general flowshop problem, whose objective function is makespan.

The main contribution of the method is going to be develop a new formulation by use of multiattribute alternative ranking. The essence of the new approach is to consider the priority of the jobs in a permutation schedule as the order of a set of ranked alternatives. The jobs are distinguished by their individual characteristics such as processing times. The most preferred job is assigned the highest rank as determined by the corresponding multiattribute value (MAV) function. The other jobs are, then, ranked according to their descending MAV function values.

To articulate the decision making process, it is supposed that the decision maker (DM), who is knowledgeable in scheduling theory, seeks for a desired ranking of the jobs so as to obtain the optimal solution with respect
to a certain objective. The scheduling theory may thus be considered as a
decision aid tool to help the DM to approach the desired goal. In fact the
DM's desire is expressed in terms of some mathematically based decision
rules. The rules helps to define and derive some unidimensional value
functions for the machines, thus yielding a MAV function for the problem at
hand.

The new approach has made it necessary to develop a proper
scheduling theory. The theoretical requirements for the general makespan
flowshop problem have been provided. The transitivity property of
multiattribute alternative ranking and the relations existing between machine
idle times and the processing times help to reduce the dimensionality of the
problem.

Finally, a heuristic method has been developed, and compared to five
well-known existing efficient algorithms. The performance of the respective
algorithms has been studied for small, medium and large problems under
different generalities and unbalanced load. The results indicate that the new
procedure is generally superior to the algorithms, especially for medium and
large problems.
In the name of Allah, the most Gracious, the most Merciful.

"And we taught him [David] the technology of making garments to protect you in your daring. Are ye then thankful?"

Holy Quran XXI: 80

To those who strive to remove underdevelopment from the world.
ACKNOWLEDGMENT

I would like to express my deepest gratitude to my advisors Prof. Lucien Duckstein and Prof. Mihajlo D. Mesarovic for their guidance, patience and encouragement throughout my studies in Case Western Reserve University. Thanks are also due to Prof. Benjamin Hobbs and Prof. Devendra Kumar for their time and effort in reading my dissertation and serving in my advisory committee. I am also deeply indebted to Prof. Jeffry Goldberg of the Systems and Industrial Engineering Department, at the University of Arizona for his review of the dissertation, much appreciated comments and helpful suggestions.

I would like to thank my friend Dr. Navid Famili at University of Akron, for his support, critique, and generosity throughout the course of this research.

This research could not be done without the dedication, patience, continuous support and encouragement of my wife Gissou who took care of our four children. I express my deepest love and appreciation to her.

Partial support for this research has been provided by REI (Ameritech Program) and the National Science Foundation (NSF) under grant No. 8802920.
TABLE OF CONTENTS

CHAPTER 1. A MULTIATTRIBUTE APPROACH TO GENERAL FLOWSHOP PROBLEMS

1.1 Introduction........................................................................................................ 1

CHAPTER 2. PROBLEM STATEMENT AND LITERATURE REVIEW

2.1 Introduction........................................................................................................ 3

2.2 Problem Statement............................................................................................. 3

   2.2.1 General deterministic flowshop problem................................................. 3
   2.2.2 Objectives of the problem.......................................................................... 5
   2.2.3 Assumptions of the problem................................................................. 7
   2.2.4 Problem formulation................................................................................. 10

2.3 Literature Review............................................................................................ 14

   2.3.1 Optimal methods.................................................................................... 14
   2.3.2 Heuristic methods.................................................................................. 16

2.4 Motivation for a new approach....................................................................... 29

2.5 Summary and Conclusions............................................................................ 30

CHAPTER 3. SCHEDULING THEORY

3.1 Introduction................................................................................................... 33

3.2 Background................................................................................................... 34

3.3 Original and reverse problems................................................................. 36

3.4 Summary and Conclusions....................................................................... 52

CHAPTER 4. MULTIATTRIBUTE APPROACH

4.1 Introduction................................................................................................... 54
4.2 General MAU formulation.............................................
4.3 MAU formulation for makespan flowshop.....................
  4.3.1 Assessment of the unidimensional value function....
  4.3.2 Assessment of the multidimensional value function..
    4.3.2.1 Additivity of MAU formulation......................
    4.3.2.2 Scaling weights of MAU formulation............... 99
4.4 Summary and Conclusions........................................

CHAPTER 5. NEW ALGORITHM AND COMPUTATIONAL RESULTS

5.1 Introduction................................................................
5.2 A new heuristic....................................................
5.3 Computational results............................................
  5.3.1 Evaluating parameter ........................................
  5.3.2 Comparison of algorithms.................................
  5.3.3 Flowshops with unbalanced load........................
  5.3.4 Comparison of MAU to NEH............................... 129
  5.3.5 Computational complexity.................................
5.4 Summary and Conclusions....................................... 132

CHAPTER 6. CONCLUSIONS AND SUGGESTIONS FOR
FUTURE WORKS

6.1 Conclusions........................................................
6.2 Suggestions for future research............................... 136
REFERENCES................................................................
APPENDICES..........................................................
CHAPTER 1

A MULTIATTRIBUTE APPROACH TO GENERAL FLOWSHOP PROBLEMS

1.1 Introduction

Sequential scheduling problems are combinatorial in nature and intractable computationally. The number of possible solutions, in general, gets astronomically large as the size of the problem increases. That is, for \( n \) jobs to be processed by \( m \) machines there exist \((n!)^m\) possible solutions. Most of the real world problems are large, having \( n > 40 \) and \( m > 40 \) for 70 percents of times (Panwalker et al., 1973). Therefore, attention has been mainly focused on efficient heuristics rather than on exact methods.

This research considers the general flowshop problem, which is a special case of the general job shop problem. This class of problems shares 20 percent of the scheduling problem in industries (Dudek et al., 1974), excluding the cases existing in administrations and institutions such as hospitals.

The main contributions of this research will be to develop a new formulation for the problem by the use of multiattribute alternative ranking, and introducing a new functional heuristic algorithm for the problem. Based on the makespan values, the frequency of obtaining better solutions and the computational effort for makespan flowshop problems, the computational results have shown that the new algorithm has been superior to the well-known existing
efficient heuristics, especially for large and unbalanced load problems. The new approach and the proposed algorithm have shown the potential of the functional approach to the problem, which has been mainly tolerated for more than two decades.

This dissertation is organized as follows. In Chapter 2 the problem definition, literature review, and motivation for the new approach are given. Chapter 3 contains the related scheduling theory to the proposed approach. In Chapter 4 the interface of multiattribute formulation and scheduling theory provides the mathematical foundations of the new approach for the problem. Chapter 5 includes a new heuristic for makespan performance and the computational results of comparing to other existing efficient procedures for various cases. A summary of the research, the conclusions, and the extensions to this study are given in the last chapter. The appendices contain the list of notation and some of the proofs.
CHAPTER 2

PROBLEM STATEMENT AND LITERATURE REVIEW

2.1 Introduction

In Section 2.2 the general flowshop problem is stated and its various types are introduced. Also, the most common objectives, assumptions, and the combinatorial optimization formulation of the problem are given. Section 2.3 contains the literature review of the exact and heuristic procedures for general makespan flowshop problems. The motivation for the new approach are declared in Section 2.5. The final section includes the summary and conclusions of the chapter.

2.2 Problem statement

2.2.1 General deterministic flowshop problem

*Flowshop* is a real industrial and administrative problem (Stary, 1982; Wiede, 1984; Karimi and Ku, 1988). Flowshop problems are a special form of *job shop* ones. In a job shop problem *n* items are to be processed by *m* processors, *m > 1*, and each job may have different processing order on the processors. That is, a vector indicating the order of the processors, *processing
vector, is assigned to each job. For instance, when job 1 is needed to be processed by processors $M_1$, $M_2$, ..., $M_n$ in a sequence, the related processing vector will be $(M_1, M_2, ..., M_n)$. The processing order is determined by the nature of the job to be processed and the operational constraints of the processors. When all the jobs have the same processing vector the problem is called flowshop, shown in Figure 2.1.

![Figure 2.1](image)

A Flowshop with $m$ machines

In general the size of the solution space for job shop problems is $(n!)^m$. For flowshop problems, by considering only the solutions having the same sequence of jobs for all the machines, the number of the possible solutions shrinks to $(n!)$.

Although the problem is found in various areas, such as hospitals and administrations, items and processor are traditionally called jobs and machines, respectively. Let $t_{ij}$ stands for the processing time of the job $i$, denoted by $J_i$, on the machine $j$, shown by $M_j$. When $t_{ij} \neq 0$, for all $i = 1, ..., n$ and $j = 1, ..., m$ the problem is called pure flowshop; otherwise, it is a general flowshop. Also, if no new job arrives to the shop until the last scheduled job be processed, it is named static flowshop; otherwise, it is a dynamic one. In case the processing times, the arrival times, and/or the availability of the machines have probability
distributions, the problem is a stochastic flowshop problem; otherwise, it is a
deterministic one. In this study we consider static deterministic general flowshop
problem, which called in brief general flowshop problem.

In general for each job $J_i$ a profile of processing times, $t_{i,j}$-profile, can be
determined. The profile for $J_i$ is constructed in the following way. The machines
are demonstrated by parallel, equidistant and equiverted lines intersecting the
horizontal axis, and the $t_{i,j}$ values are determined on the lines. The linear
piecewise graph connecting the $t_{i,j}$ value points is the $t_{i,j}$-profile of $J_i$, illustrated in Figure 2.2. Since the processing vector of a given flowshop is the
same, all the jobs can be illustrated by their $t_{i,j}$-profiles on the same coordinate.

2.2.2 Objectives of the problem

Several objectives are considered for the general flowshop problems, and
they may be conflicting. The most common objectives can be categorized based
on flow time, lateness, and setup time concepts.

Flow time of a job $J_i$ on $M_j$ is the difference between its arrival time to the
shop and its completion time by $M_j$. The flowtime of a job in a shop is the
flowtime on the last machine. Based on this concept two common objectives are
mostly considered. The first is makespan, which is the elapsed time between the
start of the earliest scheduled job and the finish of the latest scheduled one. The
second objective is the mean flow time of all the scheduled jobs. Both objectives
are to be minimized.
The next concept is lateness, which is the difference between the completion time and the due date of a job, and equals zero if job is completed by the last machine before the due date. Maximum lateness, mean lateness, and the number of the tardy jobs, which are finished after their due dates, are the corresponding objectives. In the case there exist priorities for the jobs to meet their due dates, the weighted types of these objectives are of concern, e.g., weighted mean lateness.

Setup time of a job $J_i$ on machine $M_j$ is the required time to prepare machine $M_j$ to process $J_i$ after finishing $J_{i-1}$. Total setup time of all the jobs is the most common objective in this respect, whose minimization results in both increase of machine utilization and decrease of the total elapsed time.

The above-mentioned objectives are called performance objectives or merely objectives that indicate the overall performances of the system, including
all jobs and machines. Also for any machine some certain objectives can be defined, such as an objective to minimize the idle time of a bottleneck machine or of a precious one, which do not indicate the overall performance of the systems of jobs and machines. Notice that for the bottleneck machine $M_j$ it holds (Van Wassenhove and Gelders, 1980; Fry et al., 1987).

$$\sum_{i=1}^{n} (t_{i,j}) = \max_{e=1,2,\ldots,m} \sum_{k=1}^{n} (t_{k,e})$$

We call this type of criteria partial objectives.

Although the real world problems have mostly both types of the objectives (Thuruthichara, 1984; McKay, et al., 1988; Scudder and Smith-Daniels, 1989), for the purpose of alleviating the complexity of the problem it is traditionally assumed that there exists only one performance objective for the problem. In the next section the assumptions are introduced.

2.2.3 Assumptions of the problem

The assumptions of the deterministic general flowshop problem, found extensively in the flowshop scheduling literature (Conway and Maxwell, 1967), can be put into two groups. The first includes the assumptions originated from the nature of the problem as defined, called essential assumptions. This group of assumptions are modified as the problem at hand is changed. For instance, new assumptions are needed for flexible general flowshop problems, which may have parallel machines of each type. The second group contains the simplifying
assumptions which facilitate theoretical tractability of the problem in the cost of reducing the appropriateness of the methods developed to deal with the real world problems.

The essential assumptions are as follows:
1. Each machine is persistently available to process all the scheduled jobs when it is required. That is, no shut down or break down is allowed. This assumption is valid as far as the problem at hand is considered as deterministic.
2. The order of the operations of the jobs are known.
3. Machines can be idle.
4. Each operation can be performed only by one of the machines in the shop.
5. A job can be in process at most by one of the machines at any given time.
6. There is only one machine of each type, i.e., no parallel machines are permitted.
7. All jobs are available when they are scheduled. In fact only the static general flowshop is of concern.
8. Processing time are known apriori.

The simplifying assumptions are as follows:
1. Preempting the operations is not allowed. Without this assumption the preemptable jobs may be removed from a machine before being completely processed to work on another job. Later, the preempted job is processed at the interruption point. Preemption is not feasible for some flow lines such as most chemical ones. Although the preemption is reasonable and encouraged
for some cases, for instance, when a high lateness penalty has been assigned to an arriving job to the system or to a machine, but preemption consideration causes the size of the solution space to become infinite.

2. Job cancellation is not allowed. This assumption is unreal in some cases, for instance, when a job gets major defects or damaged on a certain machine its processing should be cancelled.

3. Infinite storage space is available between machines. Clearly, this assumption is not applicable in many real shops.

4. Jobs have the same priority. This assumption is taken when flow time-based objectives are of concern, which means that all jobs are similar with respect to the lateness-based objectives. This assumption fails to be true, because in many real cases the inventory levels and/or the backorders determine the quantity and the due dates of the jobs, which are mostly different and incur different priority consideration.

5. There exists only one performance objective for the problem. This has been shown to be unrealistic (Panwalker et al., 1973; McKay et al., 1988; Scudder et al., 1989).

6. Setup time for each pair of jobs are the same. Without this assumption the problem will be a traveling salesman problem.

7. The sequence of the scheduled jobs on all the machines is the same.

Notice: A flowshop problem satisfying all the above conditions is called a traditional one.
2.2.4 Problem formulation

In this section we introduce the combinatorial optimization formulation of the general flowshop problem which considers the parameter space, solution space, objective function(s), local and global optima. The formulation indicates the inherent difficulty of the problem as a combinatorial one, and the reason of focusing on developing efficient heuristics procedures. Meanwhile it is the basis for the search methods, which will be introduced later.

Parameter space X. Parameter space is the space of the admissible data. Since different objective functions may require different data the corresponding parameter spaces may differ. For instance, problems with lateness-based objectives require priority data for the jobs but with the flowtime-based objectives do not, where the data corresponding to the processing times are required for both types of the problems. Let n and m be the number of jobs and machines, respectively. Also, let us denote

\[ T = [t_{ij}] \quad i = 1,...,n \quad ; \quad j = 1,...,m \]  \quad \text{the matrix of processing times},

\[ P = [p_i] \quad \text{where} \quad i = 1,...,n \]  \quad \text{the priority vector for the jobs}.

For makespan problem, \( X = T \) and for the simple lateness-based problems, \( X = P \times T \).

Solution space \( S_k \). As mentioned before, the required parameter space may differ according to different performance objective. Let the subscript \( k \) be used to indicate the correspondence of \( S_k \) to the performance objective \( g_k \). In general,
for a given objective $g_k$, there exist two types of solutions. The first is called **permutation schedule** that includes the solutions having the same job sequences for all machines (Conway et al., 1967). That is, a vector of size $n$ indicates the sequential schedule of the jobs. For instance, the vector $s_{k,h} = (J_{[1],k,h}, J_{[2],k,h}, \ldots, J_{[n],k,h})$, where $J_{[i],k,h}$ indicates the $i$th job in the $h$th sequence, given the objective function $g_k$. By considering all combinations of job orders, the size of the solution space is found to be $n!$.

The second is **non-permutation schedule** that indicates the solutions may have different sequences of jobs on different machines. Therefore, each solution, for a given $g_k$, will be a matrix $s_{k,h,y} = (J_{[1],k,h,y}, J_{[2],k,h,y}, \ldots, J_{[n],k,h,y})$, where, $h = 1,2,\ldots, n!$ ; $y = 1,2,\ldots,m$. Therefore, for each machine there exists a sequence of jobs of size $n$, hence, the size of the solution space will be $(n!)^m$.

By Simplifying Assumption 7, we confine our study to the permutation schedules. In fact the assumption take benefit from the similarity of the processing vectors of the jobs to reduce the solution space. This simplification is a big efficient step for reducing the computational effort in the cost of losing the non-permutation optimal schedule.

**Objective functions** $g_k$. The most common objective functions are introduced in Section 2.2.2, which can be shown as $g_k : S_k \times X \rightarrow R$, and their corresponding objective values as $C_{k,h} = g_k(s_{k,h}, x)$. Each objective function can be shown by some formula, for instance, the corresponding objective function for an $m \times n$ general makespan flowshop problem for a given sequence is
\[ T = \sum_{h=1}^{m} (t_{[i],h}) + \sum_{h=2}^{m} (I_{[i],m}) + \sum_{i=2}^{n} (t_{[i],m}) \]

where \( t_{[i],h} \) indicates the processing time of the \( i \)th scheduled job on machine \( M_h \) and \( I_{[i],m} \) indicates the idle time of the last machine, \( M_m \), waiting to process the \( i \)th scheduled job in the given sequence.

**Global and local optimal values.** For the minimizing objective function \( g_k \), let the optimal sequence and the global optimal value be \( s_k^{**} \) and \( c_k^{**} \), where \( s_k^{**} \in S_k \) and \( c_k^{**} = g_k(s_k^{**}, x) \). Then, the following relationship holds for each \( s_k \in S_k \), \( g_k(s_k^{**}, x) \leq g_k(s_k, x) \) and \( s_k^{**}, s_k \in S_k \).

For the local optimal value, first a neighborhood of the solution \( s_{k,h} \), \( h=1,2,...,n^1 \), in the solution space \( S_k \) should be determined. Let \( N(s_{k,h}) \) indicate a neighborhood around \( s_{k,h} \), which means \( N(s_{k,h}) \) encompasses all sequences that can be obtained by a local transform from \( s_{k,h} \) (Krone and Steiglitz, 1974). A local transform is a policy of interchanging the jobs in a given sequence so that to obtain other alternative sequences from the sequence. For instance, adjacent pairwise exchange of jobs is a local transform. Therefore, different local transforms may provide different neighborhoods. To differentiate various neighborhoods around \( s_{k,h} \) we use a superscript to indicate the corresponding local transform, e.g., \( N^a(s_{k,h}) \) indicates a neighborhood is constructed around \( s_{k,h} \) by the use of the local transformation \( a \).
Now, for the given objective function $g_k$, and in the neighborhood $N^a(s_{k,h})$, let the local optimal sequence and the local optimal value be $s_k^a \in N^a(s_{k,h})$ and $c_k^a = g_k(s_k^a,x)$, then, the following holds

for each $s_{k,h} \in N^a(s_{k,h})$, $g_k(s_k^a,x) \leq g_k(s_k^a,x)$

Since the solution space is finite, therefore, the existence of one (or more) global solution(s) is (are) guaranteed. However, the main obstacle which hinders finding the optimal solution is the fact neighboring points in the solution space do not necessarily correspond to neighboring points in the objective function space. This fact also prevents finding a proper local transform so that the solutions eventually converge to the global solution. There is no means to verify that a solution is globally optimal except performing an exhaustive search.

As the size of the problem increases, the size of the solution space gets astronomically large. For instance, for $n=11$, there exists $(11^3)=39,926,880$ permutation schedules, whereas in the real world 80 percent of the problems have an $n > 10$, (Panwalker et al., 1973). Therefore, applying an exhaustive search for even a modest problem is inefficient computationally, if not impossible. Practically speaking, the efficient methods are the ones that provide a non-exact solution coming as close as possible to known optimal cases or being superior to existing heuristic procedures. For this reason in the sequential scheduling area most of the attention is focused on the heuristic methods.
In the next section a literature review of the exact and heuristic methods is presented.

2.3 Literature review

The methods presented in the literature are in two groups, which are exact and heuristic. We confine our study to the makespan general makespan flowshop problem. In the next sections a very brief review of the exact methods, and a more detailed one for the heuristics will be given.

2.3.1 Optimal methods

The analytical optimal methods are confined to two- and three-machine problems, which is known as Johnson rule. Johnson (1954) introduced the optimal method for $m = 2$ and, also, for the special cases of $m = 3$, where either $\min t_{i,1} \geq \max t_{i,2}$ and/or $\min t_{i,3} \geq \max t_{i,2}$ for $i = 1,2,\ldots,n$ holds. In fact the special three-machine problem is transferred to an artificial two-machine one with the respective artificial processing times $t'_{i,1}$ and $t'_{i,2}$ for the first and the second machines,

$$t'_{i,1} = t_{i,1} + t_{i,2} \quad \text{and} \quad t'_{i,2} = t_{i,13}+ t_{i,2} \quad i = 1,2,\ldots,n.$$ 

The other optimal methods can be considered as mathematical techniques, which are briefly reviewed, as follows:
Branch and bound (BB) technique. Applying BB technique to three-machine problems in general case was studied in Ignall and Schrage (1965). The method has been widely studied in Lomnicki (1965), Brown et al. (1966), Nabeshima (1967), McMahon et al. (1967), and some others. The main achievement has been several lower bounds for the problem. Also in conjunction with the above studies some methods are developed to reduce the size of the solution space without eliminating the optimal solution, called elimination methods. Several elimination rules have been developed by Dudek and Teutron (1964), Bagga and Charkavarti (1968), McMahon (1969), Charlton and Death (1970) and Szwarc (1971,1973). The main result of these efforts is the developing of some lower bounds for the problem (Bansal, 1979). Also, some improved algorithms for job shop problems by the use of BB are provided, which can be applied to flowshop, for instance, see (Carlier and Pinson, 1989). The main drawback of this method is the increasing computational effort as the number of the job and/or the number of the machines increases.

Dynamic programing (DP). A DP formulation of the permutation makespan problem has been proposed by Nabeshima (1973). In the approach the set of sufficient conditions for determining the order of any pair of jobs to provide minimum makespan has been considered. Similar to BB, the main drawback of DP method is the increasing computational effort as the number of the job and/or the number of the machines increases.
Integer programing (IP). An IP formulation of the problem has been tried by Bowman (1953). The number of the variable is adjustable, which is at least 300 for a three-job and four-machine problem, and the constraints are much more than the number of the variables. Therefore, the method is not applicable even to modest size problems.

2.3.2 Heuristic methods

In the last three decades the efforts mainly have been focused on heuristic methods to solve general makespan problem, which can be categorized into several groups. Since some characteristics of the methods are common, a method may belong to several group, therefore it has been introduced under the more relevant one, and cited under the others. The categories are as follows.

Functional methods. These methods try to derive a function for a given general makespan flowshop problem. The jobs are then evaluated by the function and accordingly one(some) sequence(s) of jobs is (are) provided as the near optimal solution for the problem. The appealing feature of these methods is the jobs are treated separately only based on their $t_{i,j}$-profiles. This point has several remarkable advantages. First, the combinatorial problem reduces to a functional one. Second, each job can be evaluated based on their characteristics such as $t_{i,j}$-profile, which provides structural insights with respect to each individual job and also about the relationship among them. Even very complicated functional method requires less computational effort comparing to semi-exhaustive methods,
specially as the size of the problems increases. Moreover, by functionalizing approaches, a scheduled job can be omitted and/or a new job can be introduced to the shop, hence, functional methods facilitate to schedule dynamic problems. Note that in traditional flowshop problems thes dynamic cases are avoided by some simplifying assumptions.

The idea of functionalizing the flowshop problem was first proposed by Page (1961). He derived a sorting function $g$ for two-machine problems, which have been already solved optimally by Johnson (1954). The sorting function $g$ provides an index value $s_i$ for job $J_i$, which is,

$$ s_i = g(t_{i,1}, t_{i,2}) = \frac{\text{sgn}(t_{i,1} - t_{i,2})}{\min(t_{i,1}, t_{i,2})}, $$

where $\text{sgn}(t_{i,1} - t_{i,2}) = \begin{cases} 
1 & t_{i,1} > t_{i,2} \\
0 & t_{i,1} = t_{i,2} \\
-1 & t_{i,1} < t_{i,2} 
\end{cases}$ \hspace{1cm} (2.1)

Similar to the Johnson rule, Page applied the sorting function to the especial cases of three-machine problems as

$$ s_i = g(t_{i,1}, t_{i,2}, t_{i,3}) = \frac{\text{sgn}(t_{i,1} - t_{i,3})}{\min(t_{i,1} + t_{i,2}, t_{i,2} + t_{i,3})} \hspace{1cm} (2.2) $$

where $\text{sgn}(t_{i,1} - t_{i,3})$ is the same as defined in (2.1).

In fact the sorting function $g$ considers the slope sign of the line connecting $t_{i,1}$ and $t_{i,2}$ of $J_i$, and among the slopes with the same sign it considers $\min(t_{i,1}, t_{i,2})$ for sorting purpose. Finally, it ranks the jobs according to the ascending order of the $s_i$ values as given in (2.2).
Page's functional approach has been generalized for $m \times n$ flowshop problem in two ways. First, developing the idea of ranking jobs according to a slope or slope index, which has been given by Palmer (1965), Pajak (1971), and Bonney and Gundy (1976). The other was generalizing the form of the sorting function $g$, which has been introduced in Gupta (1969).

Palmer (1965) proposed a linear function $g$, mainly experimentally without any theoretical work, for the problem, which provides a slope index for job $J_i$ as

$$s_i = g\left(t_{i,1}, t_{i,2}, \ldots, t_{i,m}\right) = -\sum_{j=1}^{m} \left(m-(2j-1)/2\right) t_{i,j}, \ i = 1, 2, \ldots, n. \quad (2.3)$$

Then, jobs are sequenced according to the decreasing order of the job indices. Later Pajak (1971) has applied least square linear regression approach to points $(t_{i,j}, j), j = 1, 2, \ldots, m$, shown in Figure 2.3, and determined a slope $s_{l_i}$, for job $J_i, i = 1, 2, \ldots, n$, that is,

$$s_{l_i} = \left(\frac{m \cdot \sum_{j=1}^{m} (j^2 \cdot t_{i,j}) - \sum_{j=1}^{m} (j^2) \cdot \sum_{j=1}^{m} t_{i,j})}{(m \cdot \sum_{j=1}^{m} (j^2) - (\sum_{j=1}^{m} j)^2)}\right) \quad (2.4)$$

where $i = 1, 2, \ldots, n$.

Then, he ordered the jobs according to decreasing values of the slopes. He experimentally found that the least square regression approach provides the same order for the jobs as Palmer's method does (Bonney and Gundy, 1976). Bonney and Gundy proposed a method to improve Palmer and Pajak's ones by fitting two lines to the starting and finishing $t_{ij}$ points of $J_i$ and then using the
starting and the finishing slopes as artificial processing times $t'_{i,1}$ and $t'_{i,2}$. By this step the $m\times n$ flowshop problem will be transferred to a $2\times n$ one, and solved by Johnson rule.

![Graph](image)

Figure 2.3 The Least Square Regression Line Fitted to a $t_{i,j}$-profile

Gupta (1969) generalized Page's function to $m\times n$ problem, which is

$$s_i = g(t_{i,1}, ..., t_{i,m}) = s_j(t_{i,1} - t_{i,m}) / \min(t_{ij} + t_{ij+1})$$  \hspace{1cm} (2.5)

where $i = 1, 2, ..., n$ ; $j = 1, 2, ..., m - 1$.

Clearly, (2.5) benefits from the form of (2.2).

Several studies have compared the above cited methods either experimentally or analytically, for instance, see Bonney and Gundry (1976), Dannenberg (1977) and Gupta (1980). As shown in (2.5), Gupta's method considers at most four $t_{ij}$'s of job $J_i$, which does not provide sufficient data
points to evaluate a job especially when the number of the machines gets very large. Thus, many counterexamples can be given to show the poor performance of Gupta's method in this respect. As shown in several experimental studies Gupta's method is found to be inferior to Palmer, and similarly to Pajak, and to Bonney and Gundry methods.

The slope and slope index methods have the same drawbacks as exist in the least square linear regression, which are the existence of outlier data points and nonlinear data point pattern, which affects the estimated slopes of the data points, (Neter et al., 1985 ; Mosteller and Tuckey, 1977). The outlier in our case is a very small or very large processing time, which may affect the slopes and slope indices evaluated for each job, which in turn, may affect the final schedule. Since in a general flowshop problem the processing times may equal zero, and there is no upper limit for them, applying the cited procedures causes difficulties in such cases.

The second point, which is very important from the scheduling aspect, is the slope index of a fitted line is not a sufficient indicator to represent the processing time profile of a given job. For instance, jobs with symmetric \( t_{i,j} \) profile, i.e. having \( t_{i,j} = t_{i,m-j+1}, j=1,..., m/2 \), will have the same slope index, but, they may not have the same order in the optimal schedule. This point cannot be considered by the existing functional procedures, clearly, it will affect the resulted sequence.

In addition to the cited advantages of the functional procedures, they mostly provide inferior solutions comparing to non-functional ones. But the advantages of the functional methods such as treating each job separately,
less computation effort, and ease of applying the methods to the large
problems have been attractive features of the methods.

Generalized type. As cited in the previous category, the methods presented in
Palmer, Gupta, Pajak, and Bonney - Gundry have been some generalized features
of the functional method presented in Page (1961). Also, there exist some other
generalized methods as follows.

Johnson generalized type. The essence of these methods is to transform an
m×n flowshop problem to one or a set of 2×n artificial problems and, then,
solving the new problem(s) by Johnson rule. CDS, method developed by
Campbell, Dudeck, and Smith (1970), rapid access (RA) proposed by
Dannenberg (1977(ii)), and the method of Bonney and Gundry, cited
before, are in this type.

Based on a multiple-comparison test, called Newman-Keuls
(Newman, 1939; Keuls, 1952) quoted in Klockars and Sax (1986), which is
a powerful procedure to detect the real differences of the methods, CDS
method has been superior to RA, Gupta, Palmer and Petrov's (1966)
procedures both from performance and computational aspects, (Setiapura,
1980; Booth, 1987). CDS method provides (m-1) artificial 2×n subproblems,
which are defined as

\[ p_{i,1}^{k} = \sum_{j=1}^{k} (t_{i,j}) \quad \text{and} \quad p_{i,2}^{k} = \sum_{j=1}^{k} (t_{i,j}) \]  

\[(2.6)\]
where $p_{i,1}^k$ and $p_{i,2}^k$ represent the artificial processing times for the job $J_i$ on the first and second artificial machines in the $k$th subproblem, $k=1,2,...,m-1$. Then, the sequence with the minimum makespan value is chosen as the final solution. Besides the other drawbacks of non-functional heuristics, which will be discussed later, the major drawback of the method is the number of artificial subproblems increases as the number of the machines increases. Therefore, the computation effort increases sharply as the number of machines gets large.

Rapid access (RA) procedure utilizes a weighting system to transform the general flowshop problem to an artificial problem so that

$$p_{i,1} = \sum_{j=1}^{m} (m-j+1)(t_{i,j}) \quad \text{and} \quad p_{i,2} = \sum_{j=1}^{m} (j)(t_{i,j})$$

(2.7)

where $p_{i,1}$ and $p_{i,2}$ represent the artificial processing time for the $i$th job on the first and the second artificial machines, respectively. Later Karimi and Ku (1988) modified RA, which is called MRA. The method is very easy to apply, it is therefore being used in many real flowshops (Karimi and Ku, Ibid). The next method of this type is developed by Bonney and Gundry(1976), which has been discussed under functional methods.

The other type of generalization solves the problem for a special case, and then modifies the results for developing a procedure to deal with the general case. Petrov (1966) solved pure flowshop problem with no waiting jobs and no idle machines in the system. Then, he used the result for the traditional general
makespan flowshop. In spite of providing some structural insights, the method did not get much theoretical and practical attentions.

**Rule-based methods.** These methods are based on some rules derived from the optimal sequences either analytically or experimentally. The methods presented in King and Spachis (1980) and Newaz et al. (1983) are of this type.

King and Spachis' method focuses on minimizing the total idle time on the machines, and develops five heuristic rules for that purpose. In practice, any of the n jobs will be considered as the first job, then, the next job is chosen so as to minimize the total idle time for all the machines. Therefore, in the first step (n)(n-1) sequences are constructed and the one that provides the minimum makespan is taken and called as "chain". Each of the remaining jobs is then added to the end of the "chain". The new "chain" is the one having the minimum makespan. The procedure is repeated until all jobs are scheduled. Therefore, this method evaluates (3/2)(n)(n-1) partial sequences.

The so-called NEH procedure of Newaz et al. (1983) is based merely on one rule, which indicates that the precedence of the jobs having the largest total processing time in the optimal partial sequences is fixed. If the first partial sequences starts with the two jobs having the largest total processing times, and the procedure be continued for the next job with the largest total processing time.

The rule is applied in the following manner. First, the jobs are ordered according to the descending order of their total processing times. Second, the first two jobs are taken and all possible sequences of the jobs are constructed, and the optimal sequence is determined, and the precedence of the jobs are fixed. Then, the third
job is taken and the second step is repeated. The procedure is continued until all jobs are scheduled. This method requires \( (1/2)(n)(n+1)-1 \) number of the partial sequences to be evaluated.

Both NEH and King-Spachis' methods are semi-exhaustive and require tremendous computational effort comparing to CDS (Park, 1988). For instance, according to the experiments done on AT personal computer, for 100 problems, when the number of the jobs, \( n \), equals 20, and the number of the machines, \( m \), are 4, 5, 10, 15, 20 the average CPU times for NEH and CDS have been 120.4 and 33.05 seconds, respectively (Windmer and Hertz, 1989). Note that NEH provides closer to optimal solution than CDS. Since more than 70 percent of time both \( n > 40 \) and \( m > 40 \) (Panwalker et al., 1973), the computational effort is a very significant factor for the methods to afford the real world problems efficiently. This point requires more attention when the scheduling of a dynamic general flowshop is of concern.

For the general makespan problem rule-based methods are mostly used when some of the simplifying assumptions, which bring the problem closer to the real ones, are relaxed. More than 113 rules have been grouped by Panwalker and Iskandar (1977). Ironically, a proper combination of the rules for a certain problem itself has been an unsolved combinatorial problem (McKay et al., 1988), and some efforts are focused on this point, for example, see LaForge and Barman (1989). Since the approach is non-analytical we are not going in more detail.

Search Methods. These methods mainly focus on finding an efficient search technique to improve an initial solution. Although the initial solution is of
importance for the research effort, but it can be provided by any of the cited heuristics or even generated by random. Depending upon the search effort two types of search methods exist (Dannenberg, 1977(i)). Local search, also called close order by Dannenberg (ibid), is a one-stage search in a neighborhood N(s), which is introduced in detail in 2.2.4.(IV), to find out possible improvement in the objective function value. When the local search be continued for multiple stages, it is called extensive search.

One of the search methods is simulated annealing (SA), which is a procedure for finding good solutions to combinatorial optimization problems. The method has been proposed in Metropolis et al. (1953), and applied in Kirkpatrick et al. (1983) and Cerny (1985) to the travelling salesman problem. Later on, several authors applied the method to various combinatorial problems, (Lundy, 1985; Aarts et al., 1985; Heynderickx, 1986; Connolly, 1990; Romero and Sanchez-Flores, 1990). Ogbu and Smith (1990) have applied the method to permutation makespan flowshop problem.

The method is based on an analogy between the physical cooling of solids and combinatorial problems. The analogy has two key points. First, the different physical states of the substance corresponds to the different feasible solutions to the problem. Second, the energy of the physical system is considered as the objective function of the combinatorial problem.

SA algorithm has five steps, and the corresponding scheduling features are as follows:

1. Initialization; selecting an initial permutation sequence by any other algorithm or even by random.
2. Perturbation scheme; selecting a local transform for generating the next solution(s).

3. Evaluating objective function; determining the value(s) of the objective function for the solution(s).

4. Acceptance policy; Assigning a probabilistic acceptance policy for the evaluated solutions so that any improved solution be accepted with probability one and any other solutions with a non-zero probability.

5. Stop rule; define a stop rule for the algorithm.

Based on Step 4, SA is considered as a Markov chain problem and its asymptotical convergence with probability one to global optimal solution has been proved in Lestra (1977) and Aarts and Van Laarhoven (1985). Ogbu and Smith (Ibid.) have concluded that the performance of the algorithm significantly depends on the initial solution and also on the local transform. They applied SA algorithm to general flowshop problem, where \( n \leq 25 \) and \( m \leq 15 \), the initial solutions have been generated by Palmer and Dannenberg rapid access (RA) heuristics and the local transform has been pairwise exchange. The results indicate about 1.12 percent makespan value improvement in the cost of 25 times more computational effort.

The main difficulty in any extensive search method is the search move may return to some previous iterations, that is, the search move be entrapped in some iterations. In SA this difficulty has been tackled by assigning a probability to accept such an inferior solution. Recently, Windmer and Hertz (1989) developed a heuristic, called spirit, to tackle this difficulty. In that method they
used *Tabu search* techniques developed by Glover (1989, 1990). Tabu search provides a list of unallowed moves, called *list T*, to avoid the entrapment of the search movements in the previous iterations. The unallowed search is named *Tabu move*.

Although this approach improves the objective function value, it requires much more computational effort than NEH method. For instance, the CPU time for 100 problems of size \( n = 20, m = 4, 5, 10, 20 \), scheduled by *spirit*, NEH, and CDS require 695.24, 120.54, 33.05 seconds, respectively, (Windmer and Hertz, Ibid.). Since the computational effort gets huge for large problems, e.g., \( n > 80 \) and \( m > 80 \), these type of methods are mostly tested for problems having machines and/or jobs smaller than 20. Recently, Taillard (1990) has introduced *parallel tabu search*, which allows reduction in regular tabu search. A qualitative comparison of the methods cited in the literature review is displayed in Table 2.1

Note: A brief literature review of multiattribute ranking theory is provided in

Section 5.1
Table 2.1 The qualitative of the reviewed heuristic methods.

<table>
<thead>
<tr>
<th>Methods</th>
<th>C. S.</th>
<th>C.C.</th>
<th>Optimality</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gupta^3</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Pajak^3</td>
<td>High</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Palmer^3</td>
<td>High</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Dannen.RA^4</td>
<td>High</td>
<td>Low</td>
<td>Moderate</td>
<td>Low</td>
</tr>
<tr>
<td>Bonney-Gund.4</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Low</td>
</tr>
<tr>
<td>CDS^4</td>
<td>Moderate</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Petrov^4</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Branch&amp;Bound^5</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>DynamicProg.^5</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Integer Prog.5</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>King-Spachis^6</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>NEH^6</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Simulated Ann.6</td>
<td>Low</td>
<td>High</td>
<td>Moderate</td>
<td>Low</td>
</tr>
<tr>
<td>Spirit^6</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

1. C. S.: Computation speed
2. C.C.: Code complexity
3. Functional method.
5. Mathematical programming method.
2.4 Motivation for a new approach

Despite many theoretical studies of flowshop problem, for example, the work of Nabeshima and Szwarc partly cited in the references, there has been, in the last three decades, an increasing gap between scheduling theory and applications (McKay et al., 1988). Because of the inherent complexities of the theoretical achievements and/or the simplifying assumptions taken for the developments of the existing methods, which mostly have not been used for the real problems.

On the other hand, functional methods are welcomed in application because of ease of application, treating jobs separately, and flexibility to deal with the changes, such as the cancellation of a job and/or introducing a new job during the scheduling period. Another important advantage of the functional methods is providing some structural insights of the problem at hand. For instance, Palmer and Pajak's methods give some insight with respect to the slope of the profile of processing time for each job, which facilitates job cancellation or augment during the scheduling period. Unfortunately, these methods are either experimentally developed or are based on very simple theoretical foundations, such as least square regression approach. The lack of theoretical foundation hinders both analytical improvement for the problem and flexibility to deal with the real problems. Meanwhile, according to several researchers, they provide inferior solutions as compared to non-functional methods (Dannenberg, 1977; Gupta, 1980; Newaz, et al., 1983; Windmer and Hertz, 1989).
As shown before, even for small size problems the non-functional methods, except for CDS, require tremendous computational effort, which makes them very inefficient to be used for even modest size problems. Also the non-functional procedures do not provide structural insight of the problems, and are not apt to deal with the changing situations, as mentioned for the functional ones.

The motivation for this research aimed at providing a new approach to the problem having the following characteristics.

1. The advantages of the functional methods, such as easiness of application, apt to deal with changing situation, providing structural insight, and so on.

2. A theoretical foundation so as to help further theoretical development.

3. A smaller makespan, a higher frequency of yielding better solutions than the current functional methods, and competing with the non-functional ones in these respects.

4. Ability to consider some overall and partial objectives simultaneously.

Therefore, based on the research done, it has been found that the multiattribute value (MAV) theory can be a strong tool to provide such a procedure for the flowshop scheduling problem. The next chapter presents scheduling theory as a foundation needed for an MAV modeling of the problem.

2.5 Summary and conclusions

2.5.1 Summary
In this chapter the problem of concern has been stated, which is a static deterministic general flowshop. The most common objectives are grouped into three categories, which are flow time-based, lateness time-based, and setup time-based. The objective in this study is makespan, which is a flow time-based one. The assumptions of the problems are put into two groups, which are "essential" and "simplifying". The first is originated from the nature of the problem, and the latter is meant to reduce the complexities of the problem and facilitate solving procedure. A combinatorial optimization formulation of the problem is provided to illustrate the inherent difficulties of optimal value seeking, which is also the basis for the search methods.

A literature review of the methods is provided, mostly focused on the heuristics. The heuristics are grouped into five types, which are functional, generalized, rule-based, branch and bound, and search methods. Finally, the motivating points for a new approach are stated.

2.5.2 Conclusions

In this chapter we have found that the only way for finding the optimal value of the general makespan flowshop is an exhaustive search, which requires tremendous computational effort even for small problems. Therefore, the attention is focused on efficient heuristics. In general the heuristics can be grouped into two groups, which are a functional and a non-functional group. The functional procedures, comparing to the non-functional ones, have several advantages such as treating the jobs
separately, providing structural insights about the jobs, ease of
computation, and reducing some simplifying assumptions; however, they
provide inferior solutions. The main motivation of this study is to provide
a new functional procedure having the advantages of the functional
methods, and also compete with the non-functional ones in performance.
CHAPTER 3

SCHEDULING THEORY

3.1 Introduction

As it will be discussed later the decision maker (DM) is supposed to be familiar with scheduling theory, which works as a decision aid tool and relates multiattribute theory to the flowshop problem. In this chapter we introduce some of the required theoretical background that will be extensively used for the multiattribute formulation of the problem, given in detail in the next chapter.

Since the general flowshop problem is a combinatorial one, all possible cases should be considered. The cases are distinct by the number of the machines, the number of the jobs, and the values of the processing times. In terms of multiattribute theory, the various values of the processing times determine the attribute levels, and affect the optimal sequence and, finally, the optimal makespan. In this respect we consider the problem for all combinations of the processing times. Note that various combinations of processing times provide various combinations of the machine idle times. Therefore, we consider all possible cases of the idle times to cover all possible cases. For this purpose we need some theoretical background about the possible relations between idle times. For this task we introduce the reverse problem, which is a very strong but almost an ignored concept in general flowshop scheduling.

The main goal and contributions of this chapter are Theorems 3.1 and 3.2; the latter has been claimed, without proof and only for a globally optimal
solution, in Baker (1976, p.154). Recently, a special case of Theorems 3.2, which considers the globally optimal solutions of three-machine flowshop problems, has been proved in Salimian (1988) using a new graphic approach. In this chapter the theorems are proved for general n×m flowshop problems by the use of mathematical proof techniques, and it is shown that the claim is valid for non-optimal solutions as well.

Theorem 3.1 and its following corollary are used in Chapter 4 for reducing the number of unknown parameters and also for the determination of the unidimensional value functions. Therefore, this chapter provides the required scheduling theory for the development of the new formulation. Since Theorem 3.2 is not used in the dissertation, the corresponding proof is given in Appendix 1.

Although some of the lemmas, propositions, and corollaries are given to prove the cited theorems, we will see in the next chapter many of these theoretical developments have significant importance for the new formulation of the problem as well.

3.2 Background

Before presenting the relevant scheduling theory, we introduce some definitions as follows:

**Definition 3.1** Ready time \( r_i \). The point in time at which \( J_i \) is available for processing.
Note that the ready times for all the jobs in this study are assumed to be zero, which means that all jobs are available at the time of scheduling.

**Definition 3.2** *Completion time* \((c_{i,j})\). The time at which the processing of \(J_i\) on \(M_j\) is finished.

**Definition 3.3** *Flowtime* \((F_{i,j})\). The amount of time that \(J_i\) spends in the system until it has been completely processed by \(M_j\), that is, 
\[ F_{i,j} = c_{i,j} - r_i. \]

**Remarks**

1. For *static* general flowshop problem the ready times of jobs are the same and assumed equal to zero, therefore, \(F_{i,j} = c_{i,j}\).

2. Traditionally, flowtime of \(J_i\) without mentioning a certain machine indicates \(F_{i,m}\).

**Definition 3.4** *Idle time of machine* \(M_j\) for \(J_i\). \((I_{i,j})\). The amount of time \(M_j\) spent after processing \(J_{i-1}\) to process \(J_i\). Therefore, it holds that 
\[ I_{i,j} = \max \{ F_{i,j-1} - F_{i-1,j}, 0 \}. \]

Notice that \(I_{1,j} = 0, I_{i,1} = 0, i = 1,...,n; j = 1,2,...,m.\)
Definition 3.5 *Waiting time of* $J_i$ *for* $M_j$, $(W_{i,j})$. The amount of time that $J_i$ spent to start process on $M_j$ after completion on $M_{j-1}$.

Therefore it holds that
$$W_{i,j} = \max \{ F_{i-1,j} - F_{i,j-1}, 0 \}.$$ 

3.3 Original and reverse problems

Let a static general flowshop problem be given and be called it *original problem* (OP), for the case when makespan is being considered as the performance objective, the problem can be described by its processing time matrix, say $[t_{i,j}] \ i = 1,\ldots,n \ ; j = 1,\ldots,m$. We define the reverse problem as follows.

Definition 3.7 For a given *original* general flowshop problem with a processing time matrix $[t_{i,j}] \ i = 1,\ldots,n \ ; j = 1,\ldots,m$, the *reverse problem* (RP) of the original is a general flowshop problem having a processing time matrix $[t_{i,m-j+1}] \ i = 1,\ldots,n \ ; j = 1,\ldots,m$.

The following remark provides the relationship between the original and the corresponding reverse problems. Later on, in this and the next chapter we will use this concept for theoretical development regarding the idle times of the machines, and the makespan of the original problem. The lemma indicates that the reverse of a reverse problem is the original problem.
Remark 3.1 The reverse problem of a reverse general makespan flowshop problem is the original one. This point can be illustrated as follows. Let us the original problem be given by \([t_{i,j}] \), \(i = 1, \ldots, n; \ j = 1, \ldots, m\).

Then, by the Definition 3.1, the reverse problem can be constructed as \([t_{i,m - j + 1}] \), \(i = 1, \ldots, n; \ j = 1, \ldots, m\). For the reverse of the reverse problem it holds \([t_{i,m - (m - j + 1) + 1}] = [t_{i,j}]\), where \(i = 1, \ldots, n; \ j = 1, \ldots, m\).

Definition 3.8 Given a sequence \((a_1, a_2, \ldots, a_{k-1}, a_k)\), the reverse sequence of the given sequence is a sequence \((b_1, b_2, \ldots, b_{k-1}, b_k)\), where \(b_j = a_{m-j+1}, \ j = 1, \ldots, k\).

For all the cases when a problem and a sequence are given as original, then, the idle times and the makespan of the corresponding reverse problem are denoted by superscript \(R\), e.g., \(T^R\).

As mentioned before, all possible combinations of the processing times affect the idle times of the machines, and different idle times of machines provide different makespan values. In the next proposition we determine relationship between the set of processing times and the set of idle times.

Proposition 3.1 Given an \(m \times 2\) general flowshop problem with \([t_{i,j}]\), \(i = 1, 2; \ j = 1, \ldots, m\), and a sequence \((1, 2)\), which indicates job \(J_1\) precedes \(J_2\), let \(M_j\) and \(M_k\) be the first and the last machines for which the idle times of job \(J_2\) equal zero, that is, \(I_{2,j} = I_{2,k} = 0\); then,
\[(t_{2,j} + t_{2,j+1} + \ldots + t_{2,k-1}) \leq (t_{1,j+1} + t_{1,j+2} + \ldots + t_{1,k})\]

**Proof:** The general feature of \(I_{2,j} = I_{2,k} = 0\) is presented by a Gantt chart in Figure 3.1.

![Gantt Chart](image)

Figure 3.1

The Gantt chart for a typical permutation schedule in a two-job and \((k+1)\)-machine flowshop

There exist two cases for the propositions as follows:

(i) There exists no other machine between \(M_j\) and \(M_k\) for which the idle time to process \(J_2\) vanishes.

Using Definition 3.4 of idle time yields,
\[ I_{2,j+1} = \max \{0, t_{2,j} - t_{1,j+1}\} = t_{2,j} - t_{1,j+1} > 0 \quad (3.1, j+1) \]

\[ \vdots \]

\[ I_{2,k-1} = \max \{0, I_{2,k-2} + t_{2,k-2} - t_{1,k-1}\} \]

\[ = I_{2,k-2} + t_{2,k-2} - t_{1,k-1} > 0 \quad (3.1, k-1) \]

\[ I_{2,k} = \max \{0, I_{2,k-1} + t_{2,k-1} - t_{1,k}\} \]

\[ = I_{2,k-1} + t_{2,k-1} - t_{1,k} > 0 \quad (3.1, k) \]

By summing up both sides of (3.1)

\[ I_{2,k} = (t_{2,j} + t_{2,j+1} + \ldots + t_{2,k-1}) - (t_{1,j+1} + t_{1,j+2} + \ldots + t_{1,k}) \leq 0 \quad (3.2) \]

Finally,

\[ (t_{2,j} + t_{2,j+1} + \ldots + t_{2,k-1}) \leq (t_{1,j+1} + t_{1,j+2} + \ldots + t_{1,k}) \quad (3.3) \]

(ii) There exists at least one machine \( M_h \) between \( M_j \) and \( M_k \) for which the idle time to process \( J_2 \) vanishes.

Let us assume that there exists only one index \( h \) such that \( I_{2,h} = 0 \), where \( j < h < k \). Then, by the use of part (i) for the indices \( j \) and \( h \)

\[ (t_{2,j} + t_{2,j+1} + \ldots + t_{2,h-1}) \leq (t_{1,j+1} + t_{1,j+2} + \ldots + t_{1,h}) \quad (3.4) \]

and for the indices \( h \) and \( k \)
\[(t_{2,h} + t_{2,h+1} + ... + t_{2,k-1}) \leq (t_{1,h+1} + t_{1,h+2} + ... + t_{1,k}) \quad (3.5)\]

By adding both sides of (3.4) and (3.5) it results that

\[(t_{2,j} + t_{2,j+1} + ... + t_{2,k-1}) \leq (t_{1,j+1} + t_{1,j+2} + ... + t_{1,k})\]

Clearly, the procedure can be repeated for any number of the vanishing indices between \(j\) and \(k\). \[Q.E.D.\]

**Corollary 3.1** For the problem given in Proposition 3.1, let \(j\) be an index for which \(I_{2,j} = 0\) and \(I_{2,e} > 0\), \(e = j+1,...,k-1\). Then,

\[
\sum_{h=j}^{e-1} (t_{2,h}) > \sum_{h=j+1}^{e} (t_{1,h}) \quad e = j+1,...,k-1. \quad (3.6)
\]

**Proof:** If both sides of the first \(e\) number, where \(e = j+1,...,k-1\) of the inequalities of (3.1) be added up, then (3.6) is obtained. \[Q.E.D.\]

In the following propositions we determine the relations between the idle times of an original and its reverse problem. These propositions are important both for the development of the subsequent theorems in this chapter and for reducing of the size of the solution space, as discussed in the next chapter.

**Proposition 3.2** Given an \(m\times2\) general flowshop problem with \([t_{i,j}], i=1,2; j=1,...,m\), and a sequence \((1,2)\), which indicates job \(J_1\) precedes \(J_2\), if
$M_p$ is the first machine for which $I_{1, p}^{R} = 0$, and $M_{m-w+1}$ is the last machine so that $I_{2, m-w+1} = 0$, then, $w \geq p$.

(Notice that $I_{i, j}$ indicates the idle time of job $J_i$ on machine $M_j$ in the original problem, and superscript $R$ stands for the reverse problem within the reverse sequence).

**Proof:** By Definition 3.6, the processing time matrix of the reverse problem is $[t_{i, m-j+1}]$, $i = 1, 2; j = 1, \ldots, m$. The processing time matrices of the original and the reverse problems are given as follows.

\[
\begin{align*}
&\text{(OP)} & \begin{bmatrix} t_{1, 1} & \cdots & t_{1, p} & \cdots & t_{1, w} & \cdots & t_{1, m-w+1} & \cdots & t_{1, m-p+1} & \cdots & t_{1, m} \\
& & t_{2, 1} & \cdots & t_{2, p} & \cdots & t_{2, w} & \cdots & t_{2, m-w+1} & \cdots & t_{2, m-p+1} & \cdots & t_{2, m}
\end{bmatrix} \\
&\text{(RP)} & \begin{bmatrix} t_{1, m} & \cdots & t_{1, m-p+1} & \cdots & t_{1, m-w+1} & \cdots & t_{1, w} & \cdots & t_{1, p} & \cdots & t_{1, 1} \\
& t_{2, m} & \cdots & t_{2, m-p+1} & \cdots & t_{2, m-w+1} & \cdots & t_{2, w} & \cdots & t_{2, p} & \cdots & t_{2, m}
\end{bmatrix}
\end{align*}
\]

We consider all possible cases, which are as follows:

(i) $w < p$.

$w < p \Rightarrow m - w + 1 > m - p + 1$

Since $I_{2, m-w+1} = 0$ is the last idle time of the original problem which vanishes, it follows that $I_{2, h} > 0$, $h = m-w+2, \ldots, m$. Then, by Corollary 3.1
\[(t_{2,m} - w + 1 + t_{2,m} - w + 2 + \ldots + t_{2,m} - w) > (t_{1,m} - w + 2 + t_{1,j} + 2 + \ldots + t_{1,m})\]  
(3.7)

Also, since \(w < p\) and \(I_{1,p}^R = 0\) is the first idle time of the reversed problem which vanishes, it results that \(I_{1,p}^R > 0\). Again by Corollary 3.1 it holds from the reverse problem with the reversed sequence that,

\[(t_{1,m} + t_{1,m-1} + \ldots + t_{1,m} - w + 2) > (t_{2,m-1} + t_{2,m-2} + \ldots + t_{2,m} - w + 1)\]  
(3.8)

thus (3.7) contradicts (3.8).

(ii) \(w \geq p\).

\[w \geq p \Rightarrow m - w + 1 \leq m - p + 1\]

Since \(I_{2,m} - w + 1 = 0\) is the last idle time of the original problem which vanishes, and also \(I_{2,m} = 0\). Then, by Corollary 3.1

\[(t_{2,m} - w + 1 + t_{2,m} - w + 2 + \ldots + t_{2,m} - 1) > (t_{1,m} - w + 2 + t_{1,j} + 2 + \ldots + t_{1,m})\]  
(3.9)

Also, since \(l < p\) and \(I_{1,p}^R = 0\) is the first idle time of the reversed problem which vanishes, we can write for \(I_{1,l}^R\)

\[I_{1,w}^R = (t_{1,m} + \ldots + t_{1,m-p+2} + \ldots + t_{1,m-w+2}) - (t_{2,m-1} + \ldots + t_{2,m-p+2} + \ldots + t_{2,m+1})\]

(3.10)
(3.9) is consistent with the right hand side of equation (3.10). For \( w = p \), by \( I_{1,p}^R = 0 \), the right hand side of (3.10) negative which is consistent with the proposed inequality.

Q.E.D.

Proposition 3.3 Given an \( m \times 2 \) general flowshop problem with \( [t_{i,j}] \), \( i = 1,2 \); \( j = 1,\ldots,m \), and a sequence (1,2), which indicates job \( J_1 \) precedes \( J_2 \), if \( p \) be the first index for which \( I_{1,p}^R = 0 \), and \( m-w+1 \) be the last index such that \( I_{2,w+1} = 0 \), then, \( I_{1,w}^R = 0 \).

Notice that \( I_{1,j} \) indicates the idle time of job \( J_1 \) on machine \( M_j \) in the original problem, and superscript \( R \) stands for the reverse problem with the reverse sequence.

Proof: Let us illustrate the processing times matrices for the original and the reverse problems, which are

\[
\begin{bmatrix}
    t_{11} & \cdots & t_{1p} & \cdots & t_{1w} & \cdots & t_{1j} & \cdots & t_{1m-k+1} & \cdots & t_{1m-w+1} & \cdots & t_{1m-p+1} & \cdots & t_{1m} \\
    t_{21} & \cdots & t_{2p} & \cdots & t_{2w} & \cdots & t_{2j} & \cdots & t_{2m-k+1} & \cdots & t_{2m-w+1} & \cdots & t_{2m-p+1} & \cdots & t_{2m}
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
    t_{1m} & \cdots & t_{1m-j+1} & \cdots & t_{1m-w+1} & \cdots & t_{1m-k+1} & \cdots & t_{1j} & \cdots & t_{1k} & \cdots & t_{1w} & \cdots & t_{1p} & \cdots & t_{1m} \\
    t_{2m} & \cdots & t_{2m-p+1} & \cdots & t_{2m-w+1} & \cdots & t_{2m-k+1} & \cdots & t_{2j} & \cdots & t_{2k} & \cdots & t_{2w} & \cdots & t_{2p} & \cdots & t_{2m}
\end{bmatrix}
\]

Note that for \( p = 1 \) the case is trivial. Without loss of generality, let \( p > w \), then two subcases exist, as follows:
(i) There exist no index \( k \) such that \( I_{1,k}^R = 0 \) and \( p < k < w \).

By contradiction, let \( I_{1,w}^R > 0 \), then, by Corollary 3.1 it will hold for the reverse problem. We consider all possible cases, which are as follows:

\[
I_{1,1}^R = (t_{1,m-p+1} + \ldots + t_{1,m-w+2}) - (t_{2,m-p+1} + \ldots + t_{2,m-w+1}) > 0 \tag{3.11}
\]

Since \( I_{2,m-w+1} = 0 \) is the last vanishing idle time of the original problem and \( m-w+1 < m+p+1 \), then,

\[
I_{2,m-p+1} = (t_{2,m-w+1} + \ldots + t_{2,m-p}) - (t_{1,m-w+2} + \ldots + t_{1,m-p+1}) > 0 \tag{3.12}
\]

Since (3.11) and (3.12) are in contradiction, this results in \( I_{1,w}^R = 0 \).

(ii) There exist some indices \( k \) such that \( I_{1,k}^R = 0 \) and \( p < k < w \).

Let \( q \) be the last index before \( l \) such that \( I_{1,q}^R = 0 \). By contradiction, let \( I_{1,w}^R > 0 \), then, by Corollary 3.1 it will hold for the reverse problem. We consider all possible cases, which are as follows:

\[
I_{1,w}^R = (t_{1,m-q+1} + \ldots + t_{1,m-w+2}) - (t_{2,m-q+1} + \ldots + t_{2,m-w+1}) > 0 \tag{3.13}
\]
Since \( I_{m \cdot w + 1} = 0 \) is the last vanishing idle time of the original problem, an \( m \cdot q + 1 > m \cdot w + 1 \) holds, it results \( I_{m \cdot q + 1} > 0 \), i.e.,

\[
I_{m \cdot q + 1} = (t_{2, m \cdot w + 1} + \ldots + t_{2, m \cdot q}) - (t_{1, m \cdot w + 2} + \ldots + t_{1, m \cdot q + 1}) > 0
\]

(3.14)

As we see (3.13) and (3.14) contradict, which result in \( R_{1, w} = 0 \).

Q.E.D.

**Corollary 3.2** For the problem given in Proposition 3.3, if \( R_{m \cdot j + 1} = 0 \) as the last vanishing idle time, then \( I_{2, j} = 0 \).

**Proof:** By Remark 3.1 the original problem can be considered as the reverse of the reverse problem. In fact the reverse problem is considered as the new original and the original as the new reverse, which is shown in Remark 3.1. That is,

\[
I_{(new) 1, m \cdot j + 1} = R_{1, m \cdot j + 1} = 0 \quad \text{and} \quad R_{(new) 2, j} = I_{2, j} = 0.
\]

By Proposition 3.3,

\[
I_{(new) 1, m \cdot j + 1} = 0 \implies I_{(new) 2, j} = I_{2, j} = 0.
\]

Q.E.D.

**Corollary 3.3** For the problem given in Proposition 3.3, there exists at least one \( j \) and \( k \) such that \( R_{1, j} = 0 \) and/or \( I_{2, k} = 0 \).
Proof: We consider all possible cases, which are:

(i) There do not exist any $j$ and $k$ such that $I_{1,j}^R = 0$ and/or $I_{2,k} = 0$, where $j, k = 2, 3, ..., m$.

That is $I_{1,j}^R > 0$ and $I_{2,k} > 0$, where $j, k = 2, 3, ..., m$.

By Corollary 3.1

$$I_{1,m}^R = \sum_{h=2}^{m} (t_{1,h}) - \sum_{h=2}^{m-1} (t_{2,h}) > 0 \quad (3.15)$$

and

$$I_{1,m}^R = \sum_{h=2}^{m-1} (t_{2,h}) - \sum_{h=2}^{m} (t_{1,h}) > 0 \quad (3.16)$$

(ii) There do not exist any $j$ and $k$ such that $I_{1,j}^R = 0$ and $I_{2,k} = 0$, where $j, k = 2, 3, ..., m$.

By Proposition 3.3 if $I_{2,k} = 0$, then, $I_{1,m-k+1}^R = 0$. Let $k = 2, 3, ..., m-1$, then since $j = m-k+1$, one has $j = m-1, ..., 2$. Notice that by Corollary 3.1, a similar result can be reached for the reverse problem, i.e.,

$$I_{1,m-k+1}^R = 0 \implies I_{2,k} = 0.$$ 

(iii) There exists either $j$ or $k$ such that $I_{1,j}^R = 0$ or $I_{2,k} = 0$, where $j, k = 2, 3, ..., m$. 
If only $I_{2,m} = 0$, then, by Proposition 3.3

$$I_{1,m-m+1}^R = I_{1,1}^R = 0 \Rightarrow J = 1.$$  

Similarly,

If only $I_{1,m}^R = 0$, then,

$$I_{1,j}^R = 0 \text{ and } I_{2,k} = 0 \text{ and } I_{2,k} = 0,$$  

Q.E.D.

The following theorems are very important for the purpose of determining both the unidimensional and multidimensional value functions, which will be discussed in detail in Chapter 4. As cited in the beginning of this chapter, the following theorem has been claimed without any proof only for the globally optimal solution to the $m \times n$ flowshop problem in Baker (1974); later Salimian (1988) proved it for the globally optimal solution of the three-machine problem. Salimian introduced a new approach using two Gantt charts, similar to Figure 3.1. The charts are constructed for the original and reverse problems separately. In this study, we use mathematical techniques to prove the theorem in the general case. Also, we show that the relation between original and its reverse problems holds in non-optimal cases.

**Theorem 3.1** Given an $m \times 2$ general flowshop problem with $[t_{i,j}]$, $i = 1,2; j=1,...,m$, and a sequence $(1,2)$, let the makespan for the given sequence be denoted by $T$, and for the reverse problem with the reverse reverse sequence $TR$, then, $T = TR$. 
Proof: As shown in Corollary 3.2, two possible cases exist, as follows:

(i) There exists either $j$ or $k$ such that $I^R_{1,j} = 0$ or $I^R_{2,k} = 0$, where $j$, $k = 2, 3, ..., m$.

Without loss of generality let only $I^R_{2,m} = 0$, then,

$$I^R_{1,h} > 0 \quad h = 2, ..., m,$$

and

$$I^R_{1,m} = (t_{1,m} + ... + t_{1,2}) - (t_{2,m-1} + ... + t_{2,1})$$

Then, we can write for the makespan of the original problem,

$$T = \sum_{h=1}^{m} (t_{1,h}) + I^R_{2,m} + t_{2,m} = \sum_{h=1}^{m} (t_{1,h}) + t_{2,m}$$

Similarly, the makespan of the reverse problem in the reverse sequence is

$$T^R = \sum_{h=2}^{m} (t_{2,h}) + I^R_{1,m} + t_{1,1}$$

Substituting $I^R_{1,m}$ into $T^R$ yields

$$T^R = t_{1,m} + ... + t_{1,2} + t_{1,1} + t_{2,m}$$

Clearly, $T = T^R$. 
(ii) There exist at least one $j$ and one $k$ such that $I_{1,j}^R = 0$ and $I_{2,k} = 0$, where $j, k = 2, 3, ..., m$.

We consider the general case in which there exist several indices of $k$ and $h$. Let $k = m-l+1$ be the last index for which $I_{2,k}$ vanishes, and also $h = m-j+1$ be the last index for which $I_{1,h}^R = 0$. Then, by Proposition 3.3

$$I_{2,m-1+1} = 0 \Rightarrow I_{1,1}^R = 0$$

and

$$I_{1,m-j+1}^R = 0 \Rightarrow I_{2,j} = 0.$$  

Let also, without loss of generality,

$$1 \leq j \Rightarrow m-l+1 \geq m-j.$$  

Then, the corresponding makespan of the original problem will be

$$T^R = t_{1,1} + t_{1,2} + ... + t_{1,j} + ... + t_{1,m-1+1} + ... + t_{1,m} + I_{2,m} + t_{2,m}$$

Since $I_{2,m-1+1} = 0$ is the last vanishing idle time of the original, it holds that

$$I_{2,m} = (t_{2,m-1+1} + t_{2,m-1} + t_{2,m-2} + ... + t_{2,m-1}) - (t_{1,m-1+1} + t_{1,m-1+2} + t_{1,m-1+3} + ... + t_{1,m}) > 0$$

Substituting $I_{2,m}$ into $T$ yields

$$T = (t_{2,m} + ... + t_{2,m-1+1}) + (t_{1,j} + ... + t_{1,m-1+1}) + t_{1,1}$$
For the makespan of the reverse problem with reverse sequence, one finds

\[ T^R = t_{2,m} + t_{1,2} + \ldots + t_{2,m-1+1} + \ldots + t_{2,j} + \ldots + t_{2,1} + \frac{1}{m} + t_{1,1} \]

Since \( I^R_{1,m} = 0 \) is the last vanishing idle time of the reverse problem, then,

\[ I^R_{1,m} = (t_{1,j} + t_{1,j-1} + \ldots + t_{1,2}) \cdot (t_{2,j-1} + t_{2,j-2} + \ldots + t_{2,1}) \]

By substituting \( I^R_{1,m} \) into \( T^R \) we will have

\[ T^R = (t_{2,m} + \ldots + t_{2,m-w+1} + \ldots + t_{2,j}) + (t_{1,j} + t_{1,j-1} + \ldots + t_{1,2}) + t_{1,1} \]

Now we consider the value of \( T^R - T \), which is

\[ T^R - T = (t_{2,m-1} + t_{2,m-w+1} + \ldots + t_{2,j}) - (t_{1,j+1} + t_{1,j+2} + \ldots + t_{1,m-w+1}) \quad (3.17) \]

Notice that \( I_{2,j} = 0 \) and \( I_{2,m-w+1} = 0 \), and since it is assumed that \( m-j+1 \leq m-w+1 \) then, by Proposition 3.3 we can write

\[ (t_{2,j} + \ldots + t_{2,m-w+1} + t_{2,m-1}) - (t_{1,j+1} + t_{1,j+2} + \ldots + t_{1,m-w+1}) \leq 0 \]

\[ (3.18) \]

Comparing (3.17) and (3.18)

\[ T^R - T \leq 0 \quad (3.19) \]

Now we determine the value of \( T - T^R \), which is
\[ T - T^R = (t_{1, j+1} + t_{1, j+2} + \ldots + t_{1, m-1}) - (t_{2, j+1} + t_{2, m+1} + t_{2, m-1}) \]  
\[ (t_{1, m-w+1} + \ldots + t_{1, j+2} + t_{1, j+1}) - (t_{2, m-1} + t_{2, m-w+1} + \ldots + t_{2, j}) \leq 0 \]  
(3.20)

Similar to the procedure applied for the original problem, since 
\( I^R_{1,1} = 0 \) and \( I^R_{1, m-j+1} = 0 \), by Proposition 3.1

\[ (t_{1, m-w+1} + \ldots + t_{1, j+2} + t_{1, j+1}) - (t_{2, m-1} + t_{2, m-w+1} + \ldots + t_{2, j}) \leq 0 \]  
(3.21)

From (3.19) and (3.21) it results that

\[ T - T^R \leq 0 \]  
(3.22)

Finally, from (3.19) and (3.22) we conclude that \( T = T^R \). Q.E.D.

**Corollary 3.3**: For the problem given in Theorem 3.1, if the given sequence is optimal for the original problem, then, the reverse sequence is optimal for the reverse problem.

**Proof**: Let the makespans corresponding to the original problem with the optimal sequence be denoted by \( T^* \), and the makespan for the reverse problem with the reverse of the optimal sequence be denoted by \( T^{*R} \).

By Theorem 3.1, \( T^* = T^{*R} \).

By contradiction, let \( T^{*R} \) be nonoptimal, that is, there exists \( T^{+R} \) such that \( T^{*R} > T^{+R} \). Then, by Theorem 3.1, \( T^{+R} = T^* \), where \( T^* \) is the corresponding makespan of the optimal reverse problem.

Finally, it results that \( T^* = T^{+R} > T^{+R} \), which contradicts the
optimality of $T^*$. Thus, the reverse of the optimal sequence is the optimal sequence of the reverse problem. Q.E.D.

In the next theorem we generalize Theorem 3.1 to the $m \times n$ case.

**Theorem 3.2** Given an $m \times n$ general flowshop problem with $[t_{ij}, j], i = 1, \ldots, n$; $j = 1, \ldots, m$, and a sequence $(1, 2, \ldots, i, i+1, \ldots, n)$, which indicates that the sequence of the jobs, e.g., $J_i$ precedes $J_{i+1}$. Let the makespan for the given sequence be denoted by $T$, and for the reverse problem with the reverse sequence $T^R$, then, $T = T^R$.

**Proof**: The proof is given in Appendix 1.

### 3.4 Summary and conclusions

#### 3.4.1 Summary

In this chapter we have developed the scheduling theory that is required for the multiattribute formulation of the general makespan flowshop problem. For this purpose the concepts of the original and the reverse flowshop problems are introduced. The relations between the original problem and the corresponding reverse one facilitate the derivation of essential propositions. The propositions provide tools to consider all combinations of machines idle times instead of considering all combinations of job processing times.
3.4.2 Conclusions

In this chapter it has been shown that all possible combinations of the processing times affect machine idle times. The relations between the processing times and idle times of the machines can be determined by considering the relations between vanishing and non-vanishing idle times. Also, some relations between the idle times of the original and the reverse problems have been found, which are used to develop some basic theorems. For general makespan flowshop problems the makespan of the original problem with a given sequence is equal to the makespan of the reverse problem in the reverse of the given sequence. This has been proved for the $m \times 2$ problem in Theorem 3.1, and generalized for the $m \times n$ one in Theorem 3.2.
CHAPTER 4

MULTIATTRIBUTE APPROACH

4.1 Introduction

This chapter introduces a new functional procedure to handle the general flowshop problem by the use of multiattribute value (MAV) theory. The essence of this approach is to consider the sequential scheduling problem as a multiattribute alternative ranking one. Multiattribute alternative ranking is used as a decision making aid for problems having conflicting objectives. Such problems can be found in various areas, for instance, in water resource planning and development. A large body of the multiattribute literature focuses on the related techniques and case studies, for a bibliography see Siskos et al. (1984(i)). We present a brief review of three multicriterion methods, which have extensively been applied to various cases.

The first method is based on multiattribute utility theory (Keeney, 1974), which assesses a unidimensional utility function for each attribute separately. Then, by assuming the preferential and utility independence a multiattribute utility function for the problem at hand may be estimated. The multiattribute function provides an index, which determines the level of the desirability of each alternative. The alternatives can thus be ranked
from the most to the least desirable one. A detailed case study is given in Keeney and Wood (1977).

The second procedure, called ELECTRE I, has been developed by Benayoun et al. (1966) and Roy (1971). The basic idea in this method is to find those alternatives that are preferred for most of the attributes, and do not provide an unacceptable level of discontent for the other attributes. The method requires three concepts, which are "concordance" and "discordance" and their corresponding "thresholds values". The decision maker determines the relative importance of the attributes by assigning a weight for each attribute. The thresholds are also selected with the help of the decision maker. The result of applying ELECTRE I to a set of alternatives is a preference graph, which indicates a partial ranking of the alternatives. A complete ranking of the alternative may be provided by ELECTRE II.

ELECTRE II, developed in Roy and Bertier (1971), is an outranking method, which is based on the same idea of ELECTRE I. Three levels of the concordance and discordance are articulated for the method, called low, average, and high. The levels are determined by three intervals for each concordance, discordance, and threshold values. Then, the preference relation between each pair of the alternatives is defined according to the corresponding levels of concordance and discordance indices. Depending upon the concordance and discordance levels there exists either a weak preference or a strong one. The preference relations are used to provide two preference graphs, which can be used to yield a complete ordering.
ELECTRE I has been applied to several case studies in David and Duckstein (1976), Diaz-Pena et al.(1978), and Gershon et al.(1982). Some case studies of ELECTRE II can be found in Gershon et al. (Ibid.) and Duckstein and Gershon (1983). Later the method has been extended for structural analysis of the relation among the alternatives (Duckstein and Kempf, 1979) and to the problem with uncertainty (Siskos et al., 1984(ii)). The concordance and discordance analysis without using a graph is also presented in Nijkamp and Vos (1977).

The last approach is compromise programming, which uses the ideal and anti-ideal solutions, including infeasible ones. Then, by the use of a distance criterion, the method tries to find a solution as close as possible to the ideal and/or to as far as possible to the anti-ideal solution (Zeleny, 1973), (Yoon, 1988). A case study can be found in Duckstein and Opricovic (1980).

There are some theoretical efforts to enrich the mathematical foundation of multiattribute ranking theory. Although not applied to real case problems, the two recently developed procedures, called MAPPAC and PRAGMA, in Matarazzo (1986, 1988) deal with multiattribute alternative ranking. The basic idea in these procedures is to consider the partial and global profiles of the alternatives, and obtain the relevant weighted sum of the partial ranking frequencies for each alternative, called the global frequency for the alternative. PRAGAMA provides both a partial and a global ranking of the alternatives. As a secondary result, a partial and overall insight of the structure of the system of the alternatives
can be obtained. For instance, from sequential scheduling aspect, PRAGMA yields the sequence of jobs in the intermediate queue of each pair of the consequent machines, meanwhile a permutation sequence is also produced. In this part, it is worthwhile to see Bouyssou's works.

Before discussing the method used in this study we introduce the difference between utility theory and value theory. The preference representation function under uncertainty is called a *utility function*, and under certainty is a *value function*. In fact, multiattribute utility theory deals with the cases having risk (Keeney, 1974; Keeney and Raiffa, 1976), and multiattribute value theory considers the deterministic cases (Dyer and Sarin, 1979).

In this research we apply the multiattribute value theory to provide a ranking procedure for general makespan flowshop problems. Jobs are considered as a set of alternatives, which are distinguished by their processing times and other individual characteristics such as priority weights for processing. In each MAV ranking problem the decision maker (DM), the objective functions, and the corresponding unidimensional utility or value functions, and their attribute levels should be identified, which are introduced as follows.

It is assumed that the DM, who seeks the optimal solution for a given performance objective and the scheduling analyst is the same person. That is, the DM is knowledgeable of sequential scheduling theory. It is possible to show that the DM as a scheduler can benefit from the scheduling theory, developed in Chapter 3, to introduce a unidimensional
value function for each machine. The parameters defined in Section 2.2.4 determine the corresponding attributes levels. By determining a MAV function for the scheduling problem, it can be considered as a multiattribute alternative ranking one. That is, the jobs are ordered according to their descending MAV function values, which determine the priority of the jobs to be processed.

To articulate the decision making process, there exist two different levels of the courses of actions for the DM. In the first level, he seeks the most desirable ranking, or equivalently sequence, of the jobs with regard to a given performance objective function. In this level, the scheduling theory assists the DM by providing mathematically based decision rules. In fact scheduling theory works as a decision aid to the DM for job sequencing. The rules are called optimal rules, which represent the optimal seeking desire of the DM, and they provide a benchmark for the next level.

In the next level some partial and overall objectives may exist, and/or some constraints on the jobs and/or on the machines may have to be considered. For instance, certain priority levels have to be assigned for a set of jobs, and idle time consideration has to be considered for certain machines. The DM can benefit from scheduling theory to generate and select the set of desirable solutions by modifying the uni- or multi-dimensional value functions.

In Section 2 a general MAV formulation of permutation general flowshop problems is presented. The MAV formulation to the problem with makespan as the performance objective, which includes the
assessment of the uni- and multi-dimensional value functions, is given in Section 3.

4.2 General MAV formulation

For the general flowshop problem with the performance objective \( g_k \), let the *order-preserving unidimensional value function*, or briefly called *value function*, of machine \( M_j \) be denoted by \( V_{k,j}(.) \). Since the set of the possible solutions is finite the existence of the value function is guaranteed (Fishburn, 1970 as quoted in Szidarovsky et al., 1980). The levels of attributes that relate the job \( J_i \) to the machine \( M_j \) can be represented by a function \( f_{k,i,j}(.) \) of the parameter \( x_{i,j} \), given in Section 2.2.4., where \( x_{i,j} \in X \) and \( X \) is the admissible data set. We call \( f_{k,i,j}(.) \) as *parameter function*, which for a given performance function \( g_k \) provides the attribute level related to \( M_j \) with respect to \( J_i \). From scheduling theory aspect, in general, the parameter function \( f_{k,i,j}(.) \) facilitates to consider a variety of relations between a subset of the jobs and a subset of the machines, for instance, between the subset of the high priority jobs and/or precious and the subset of bottleneck machines. By the definition of *preference* in value theory (Dyer and Sarin, Ibid.), for a given \( g_k \) for each two jobs \( J_h \) and \( J_i \), it holds that

\( J_i \) is preferred to \( J_h \) with respect to \( M_j \), if and only if

\[
V_{k,j}(f_{k,i,j}(x_{i,j})) > V_{k,j}(f_{k,h,j}(x_{h,j}))
\]  

(4.1)
The MAV function value for job $J_i$ with respect to the performance objective $k$ is denoted by

$$V_k(f_{k,i,j}(x_{i,j})) = V_k(f_{k,i,1}(x_{i,1}), f_{k,i,2}(x_{i,2}), \ldots, f_{k,i,m}(x_{i,m}))$$ (4.2)

and $J_h$ is preferred to $J_1$, with respect to $g_k$ if and only if

$$V_k(f_{k,h,1}(x_{i,1}), f_{k,h,2}(x_{i,2}), \ldots, f_{k,h,m}(x_{i,m})) < V_k(f_{k,i,1}(x_{i,1}), f_{k,i,2}(x_{i,2}), \ldots, f_{k,i,m}(x_{i,m}))$$ (4.3)

One of the advantages of the MAV approach is to benefit from the transitive property of alternative preference, that is, if for a performance objective $g_k$ if $J_i$ is preferred to $J_h$, and $J_h$ is preferred to $J_1$, then, $J_h$ is preferred to $J_1$.

This property enables the study of the optimal conditions of general $m \times n$ flowshop problems by analyzing an $m \times 2$ one, instead.

### 4.3 MAV formulation for makespan flowshop

#### 4.3.1 Assessment of the unidimensional value function

The general formulation, which is given in the last section, is applied to the makespan problem. Since only one performance objective exists we
omit the subscript k from \( f_{k,l}(.) \) and \( V_{k,l}(.) \) functions. That is, \( f_{l}(.) \) and \( V_{l}(.) \) are the parameter function and the unidimensional value function of the processing times of the jobs, respectively. In fact we apply the first level of the decision making process, where there are no other partial and overall objectives.

Given a sequence, the makespan of a set of jobs directly depends on the processing times of the jobs. Hence, we can determine the parameter function \( f_{l}(.) \), introduced in 4.2, as an identity function. It is worthwhile to notice that for the case of having partial objectives and/or due date-based objectives the function may not be an identity one.

The following propositions provide some characteristics of the unidimensional value functions for the machines in a general makespan flowshop problem. The value functions are derived with assuming that all other attributes are held fixed at any specified level (Dyer and Sarin, Ibid.).

As already mentioned the processing times of the jobs on the machines, \( t_{i,j} \), determine the levels of the attributes. If all the processing times of the jobs are fixed except for a certain machine \( M_{j} \), then \( t_{i,j} \), \( i=1,...,n \), are the only factors that differentiate one job from another. Therefore, for a makespan flowshop a change in the processing time of job \( J_{k} \) on machine \( M_{j} \), \( t_{k,j} \), may change the order of \( J_{k} \) in the optimal permutation sequence.

Proposition 4.1 indicates that to obtain a sequence of jobs for a given general makespan problem \( V_{1}(.) \) of \( t_{i,1} \) has to be a decreasing function. In fact the position of a job in the optimal sequence is considered as an index,
which is determined by its level of the value function. To implement the idea, we assume that processing times of the jobs on all machines except for the first one are fixed. Then, we consider the characteristics of $V_1(.)$ as a function of $t_{i,1}$ to get the minimum makespan.

**Proposition 4.1** For an $m \times n$ general makespan flowshop problem, the unidimensional value function for the first machine, $V_1(.)$, is a decreasing function of the processing times of $t_{i,1}$, $i = 1, 2, ..., n$.

**Proof:** We choose two jobs $J_h$ and $J_i$ such that $t_{h,1} < t_{i,1}$, where $h \neq i$, and $h, i = 1, 2, ..., n$. Two permutation schedules can be made by the jobs, which are $(h, i)$ and $(i, h)$. As shown in Figure 4.1, let the makespans and the idle times of the sequences $(h, i)$ and $(i, h)$ be denoted by $T_h, t_h^h, i_k^h, j_k, k = 1, ..., n$, respectively.

![Gantt Chart](image)

**Figure 4.1**

The Gantt Chart for the Sequence $(h, i)$
As cited in Section 4.3.1 the attributes, or equivalently the processing time, may take any value that are fixed for each machine. That is,
\[ t_{h,j} = t_{i,j}, \quad j=2,\ldots,m \quad (4.4) \]
We should consider the relationship between \( T_h \), and \( T_i \) for all possible conditions, where
\[
T_h = \sum_{j=1}^{m-1} (t_{h,j}) + t_{h,m} + t_{i,m} \quad (4.5)
\]
and
\[
T_i = \sum_{j=1}^{m-1} (t_{i,j}) + t_{i,m} + t_{h,m} \quad (4.6)
\]

The makespan values depend on the processing time the values and the last machine idle times. The last machine idle times, according to Definition 3.3, change as the job sequence and/or the processing time values alter. Hence, all possible cases of the processing times provide various combination of idle times. Therefore, to compare \( T_h \) and \( T_i \) for all possible cases, we must cover all possible values of the corresponding \( l_{h,j,m} \) and \( l_{i,h,m} \). We remind that by Definition 3.4,
\[
l_{h,i,1} = 0 \quad (4.7)
\]
and
\[ I^h_{i,j} = \max \{ 0, I^h_{i,j-1} + t_{i,j-1} - t_{i,j} \} \quad j = 2,3,\ldots,m \] (4.8)

and

\[ I^i_{h,1} = 0 \] (4.9)

and

\[ I^i_{h,j} = \max \{ 0, I^i_{h,k-1} + t_{h,j-1} - t_{i,j} \} \quad j = 2,3,\ldots,m \] (4.10)

The possible relationships between \( I^i_{h,j} \) and \( I^h_{i,j} \), \( j = 2,3,\ldots,m \), are as follows:

(i) \( I^h_{i,j} = I^i_{h,j} = 0 \) for at least one \( j \).

In this case, let us consider the first \( j \) for which \( I^h_{i,j} = I^i_{h,j} = 0 \), then, from (4.8) and (4.10)

\[ I^i_{h,j+1} = \max \{ 0, I^i_{h,j} + t_{h,j} - t_{i,j+1} \} \] (4.11)

where \( I^i_{h,j} = 0 \)

and

\[ I^h_{i,j+1} = \max \{ 0, I^h_{i,j} + t_{j} - t_{h,j+1} \} \] (4.12)

where \( I^h_{i,j} = 0 \).

By (4.4), (4.11) and (4.12)

\[ I^h_{i,m} = I^i_{h,m} \] (4.13)
Since it is assumed \( t_{h,1} < t_{i,1} \), where \( h \neq i \), then by (4.4), (4.5) and (4.6) it results

\[ T_h < T_i \]

(ii) \( I^h_{i,j} > 0 \) \( j = 2, \ldots, m \), and \( I^h_{1,k} = 0 \) for at least one \( k \), \( k = 2, \ldots, m \).

By contradiction, let us consider the first \( k \) for which \( I^h_{1,k} = 0 \).

By the idle time equations given in (4.8) and (4.10), we have

\[ I^i_{h,k} > 0 \quad \Rightarrow \quad t_{h,1} > t_{i,k} \quad (4.14) \]

and

\[ I^i_{j} = 0 \quad \Rightarrow \quad t_{i,1} \leq t_{i,k} \quad (4.15) \]

since \( t_{h,k} \leq t_{i,k} \), it yields

\[ t_{i,1} > t_{h,1} \] which contradicts the assumption.

(ii) \( I^h_{1,k} > 0 \) \( k = 2, \ldots, m \), and \( I^h_{1,j} = 0 \) for at least one \( j \), \( j = 2, \ldots, m \).

Let the first \( j \) for which \( I^h_{1,j} = 0 \) be \( j' \), then,

\[ I^h_{1,j'} = \max \{ 0, \ t_{h,1} - t_{i,j'} \} \quad \Rightarrow \quad t_{h,1} \leq t_{i,j'} \quad (4.16) \]

and
\[ l_{i,j}^1 = \max \{ 0, t_{i,1} - t_{h,j}^* \} \quad \Rightarrow \quad t_{i,1} > t_{i,j}^* \]

Also, let the next \( j \) for which \( l_{h,j}^1 = 0 \) be \( j'' \), similar to (4.16)

\[ l_{i,j}^1 = 0 \quad \Rightarrow \quad t_{h,j}^* = t_{i,j}^* \]

(4.17)

Since \( t_{h,j}^* = t_{i,j}^* \), from (4.16) and (4.17)

\[ t_{h,1} \leq t_{i,j}^* \]

(4.18)

and

\[ l_{i,j}^h > 0 \quad \Rightarrow \quad t_{i,1} > t_{h,j}^* \]

We continue the procedure to the last, say \( q^\text{th}, j = j^{(q)} \) for which \( l_{h,j}^1 = 0 \). Clearly, as concluded in (4.17) and (4.18), it can be written as

\[ l_{i,j}^{(q)} = 0 \quad \Rightarrow \quad t_{h,1} \leq t_{i,j}^{(q)} \]

(4.19)

and

\[ l_{i,j}^{(q)} > 0 \quad \Rightarrow \quad t_{i,1} \leq t_{h,j}^{(q)} \]

(4.20)

By the use of (4.7) and (4.20)

\[ l_{i,m}^h = t_{h,j}^{(q)} - t_{i,m} \]

(4.21)

and by (4.8) and (4.19)

\[ l_{i,m}^h = t_{i,1} - t_{h,m} \]

(4.22)

By substituting the values of \( l_{i,m}^h \) and \( l_{i,m}^1 \) in (4.5) and (4.6),
\[ T_h = t_{h,1} + \sum_{j=1}^{m-1} (t_{h,j} + 0 + (t_{h,j} - t_{i,m}) + t_{h,m} + t_{h,m} \]

and

\[ T_i = t_{i,1} + \sum_{j=1}^{m-1} (t_{i,j} + 0 + (t_{i,1} - t_{h,m}) + t_{i,m} + t_{i,m} \]

=> \( T_h < T_i \)

(iv) \( I_{h,k}^l = 0 \) \( k=2,\ldots,m \), and \( I_{h,j}^l = 0 \) for some \( j \) and \( k \), where \( j \neq k \), and \( k, j = 2,\ldots,m \). Several subcases have to be considered, which are:

- \( k < j \). By the argument given in (ii) this case is not possible.
- \( k > j \). We showed that \( I_{h,k}^l = 0 \) must hold. Otherwise, \( I_{h,j}^l > 0 \).

Let the last index before \( k \) for which \( I_{h,j}^l \) is vanished be \( j' \), by

the similar argument given for (iii) it is concluded that

\[ t_{h,j'} < t_{i,1} \]  \hspace{1cm} (4.23)

and also

\[ I_{h,j'+1}^l = \max \{0, t_{h,j'} - t_{i,j'+1} \} \]

and

\[ I_{h,j'+1}^l = \max \{0, t_{i,1} - t_{h,j'+1} \} \]
Since \( P_{i,k} = 0 \) \( \Rightarrow \) \( t_{i,1} \leq t_{h,k} \)

and

since \( I_{i,j} > 0 \) \( \Rightarrow \) \( t_{h,j} > t_{i,k} \)

Then, by \( t_{h,k} = t_{i,k} \) \( \Rightarrow \) \( t_{h,j} > t_{i,1} \) which contradicts (4.23)

\( \Rightarrow \) \( I_{h,k} = 0. \)

Now the case is the same as (i), thus, \( T_{h} < T_{i}. \)

As illustrated in the above cases, by noticing that makespan is a minimizing objective, \( T_{h} \) is preferred to \( T_{i} \), and so is the sequence \((h,i)\) to \((i,h)\), That is,

\( t_{h,1} < t_{i,1} \) \( \Rightarrow \) \( T_{h} \leq T_{i} \) \( \Rightarrow \) \( (h,i) \) is preferred to \((i,h)\)

\( \Rightarrow \) \( J_{h} \) is preferred to \( J_{i} \)

\( \Rightarrow \) \( V_i(t_{h,1}) \geq V_i(t_{i,1}) \) \( i \neq h \); \( i,h = 1,2,\ldots,n. \) (4.24)

Q.E.D.

The next proposition considers the corresponding unidimensional value function for the last machine, as a function of the processing times i.e. \( V_m(t_{h,m}), h = 1,\ldots,n. \)
Proposition 4.2 For an \( m \times n \) general makespan flowshop problem, the unidimensional value function for the last machine, \( V_{m}(\cdot) \), \( h=1,\ldots,n \). is an increasing function of the processing times \( t_{i,m} \), \( i = 1,2,\ldots,n \).

**Proof:** Choose two jobs \( J_h \) and \( J_i \) such \( t_{h,m} > t_{i,m} \), where \( h = i \), and \( h, i = 1,2,\ldots,m \). As shown in Figure 4.1 two permutation schedules could be made by the jobs, which are \((h,i)\) and \((i,h)\). Let the makespans and the idle times of the sequences \((h,i)\) and \((i,h)\) be denoted by \( T_h, I^h \), and \( T_i, I^i \), respectively.

As mentioned before, for the purpose of determining the effect of various attribute levels on the makespans corresponding to the sequences, the attribute levels of each machine except for the last one are fixed. That is,

\[
t_{k,j} = t_{i,m}, \quad j = 1,\ldots,m-1
\]  

(4.25)

By (4.25), (4.7) and (7.8), for any value of \( t_{k,j} = t_{i,m}, j=1,\ldots,m-1 \), it could be written:

\[
I_{h,m-1}^i = I_{h,m-1}^i
\]  

(4.26)

The cases that may affect the corresponding makespans are as follows:

(i) \( I_{h,m}^i > 0 \) and \( I_{h,m}^i > 0 \).
By the definition of makespan, which is the flowtime on the last machine,

\[ T_h = \sum_{j=1}^{m-1} (t_{h,j}) + I_{i,m}^h + t_{i,m} + t_{h,m} \]  

\[(4.27)\]

and

\[ T_i = \sum_{j=1}^{m-1} (t_{i,j}) + I_{i,m}^i + t_{i,m} + t_{h,m} \]  

\[(4.28)\]

Since \( t_{h,m} > t_{i,m} \) is assumed, then, by (4.25) and (4.26)

\[ \Rightarrow \quad T_i < T_h \]

(ii) \( I_{i,m}^i = 0 \) and \( I_{i,m}^h > 0 \).

By definition

\[ I_{i,m}^h = \max \{0, I_{i,m-1}^h + t_{i,m-1} - t_{h,m}\} > 0 \]  

\[(4.29)\]

\[ \Rightarrow \quad I_{i,m-1}^h + t_{i,m-1} > t_{h,m} \]  

\[(4.30)\]

and similarly,

\[ I_{i,m}^i = 0 \quad \Rightarrow \quad I_{i,m-1}^i + t_{i,m-1} < t_{i,m} \]  

\[(4.31)\]

(4.31) contradicts (4.26).

(iii) \( I_{i,m}^i > 0 \) and \( I_{i,m}^h = 0 \).
By \( I_{h,m}^i = 0 \) \( \Rightarrow \) \( I_{h,m-1}^i + t_{i,m-1} \leq t_{h,m} \) (4.32)

\[
T_h = \sum_{j=1}^{m-1} (t_{h,j}) + t_{h,m} + t_{i,m} \tag{4.33}
\]

and

\[
T_i = \sum_{j=1}^{m-1} (t_{i,j}) + I_{h,m-1}^i + t_{h,m-1} + t_{h,m} \tag{4.34}
\]

Then, by (4.32), (4.33) and (4.34) \( \Rightarrow \) \( T_i < T_h \)

(iv) \( I_{h,m}^i = 0 \) and \( I_{h,m}^i = 0 \).

Using (4.5) and (4.6) \( \Rightarrow \) \( T_i < T_h \)

As illustrated in the above cases, noticing that makespan is a minimizing objective, \( T_i \) is preferred to \( T_h \), and so is the sequence \((i,h)\) to \((h,i)\). That is,

\[
t_{h,1} > t_{i,1} \quad \Rightarrow \quad T_h \geq T_i \quad \Rightarrow \quad (i,h) \text{ is preferred to } (h,i) \quad \Rightarrow
\]

\[
\Rightarrow \quad V_m(t_{h,1}) \geq V_m(t_{i,1}) \quad i \neq h ; \quad i,h = 1,2,...,n. \quad \text{Q.E.D.}
\]

Propositions 4.1 and 4.2 provide some insights on the characteristics of the first and the last machines. As we see the machines are in symmetric position
with respect to the middle machine(s). In the following proposition we show that
the characteristics of the unidimensional value functions are similarly valid for
other symmetric machines in the shop. That is, for the machine $M_j$ the
unidimensional value function is decreasing provided that the value function is an
increasing function of the processing time of the machine $M_{m-j+1}$, $j = 1, 2, ..., \lfloor m/2 \rfloor$. (Notice that $\lfloor m/2 \rfloor = m/2$ when $m$ is even, otherwise, it equals $m/2-1$).

In the proposition we consider all possible values of the processing
times, which determines the attribute values, through considering all
possible values of the idle times. Meanwhile we benefit from the properties
of the reverse problem with the reverse sequence, introduced in the last
chapter. So first, all attributes except an arbitrarily selected pair are fixed,
then, all possibilities of the last vanishing idle times are considered.
Depending upon the employment of the original or the reverse problem, to
prove the proposition either of the attributes is fixed. Also, we take
advantage from the relationships between the first and the last vanishing
idle times, given in Chapter 3.

**Proposition 4.3** For an $m \times n$ general makespan flowshop problem, if the
unidimensional value function for the machine $M_{m-j+1}$ is an
increasing function of $t_{i,m-j+1}$, $i = 1, 2, ..., n$, then, the
unidimensional value function for the machine $M_j$, $j = 1, 2, ..., \lfloor m/2 \rfloor$, is a decreasing function of the processing time
\[ t_{i,1}, \ i = 1,2,\ldots,n; \ (\lfloor m/2 \rfloor = m/2 \text{ when } m \text{ is even, otherwise, it equals } m/2-1). \]

**Proof:** Let us consider two jobs \( J_h \) and \( J_i \), \( h \neq i, h, i = 1,2,\ldots,m \), such that \( t_{i,e} = t_{h,e} \) where \( e \neq j \), \( m-j+1, e = 1,2,\ldots,m \). The processing time matrix of the jobs is

\[
\begin{bmatrix}
  t_1 & t_2 & \cdots & t_h & \cdots & t_h & m-j+1 & \cdots & t_m \\
  t_1 & t_2 & \cdots & t_i & \cdots & t_i & m-j+1 & \cdots & t_m
\end{bmatrix}
\]

Two permutation schedules can be made by the jobs, which are \((h,i)\), and \((i,h)\). Let the makespans and the idle times of the sequences \((h,i)\) and \((i,h)\) be denoted by \( T_h \), \( I^h \), and \( T_i \), \( I^i \), respectively. Let the first vanishing index after \( j \) and the last vanishing one for \( I^h_{i,j} \) be \( p' \) and \( p \), respectively, and similarly \( q' \) and \( q \) for \( I^i_{h,i} \). Let \( t_{i,e} = t_{h,e} \) where \( e \neq j \), \( m-j+1, e = 1,2,\ldots,m \).

There are several cases as follows:

(i) \( p \leq j \) and \( q \leq j \)

Since \( t_{i,e} = t_{h,e} \) where \( e \neq j \), \( m-j+1, e = 1,2,\ldots,j-1 \), it results in \( p = q \) then,

\[ I^h_{i,m} = t_{i,1} + t_{i,j} + t_{i,m-j+1} - t_{h,j} - t_{h,m-j+1} - t_{h,m} \]

and
\[ I_{h,m}^i = t_{h,1} + t_{h,j} + t_{h,m-j+1} - t_{i,j} - t_{i,m-j+1} - t_{i,m} \]

Let fix \( t_{i,m-j+1} = t_{h,m-j+1} \), and let \( t_{h,j} < t_{i,j} \), then, by the use of the makespan formula, given in (4.35) and (4.36)

\[ T_i - T_h = t_{i,j} + (t_{h,j} - t_{i,m}) - t_{h,j} - (t_{i,j} - t_{h,m}) = 0 \quad (4.35) \]

\[ => T_i = T_h \]

Since \( T_i = T_h \) indicates that \( J_i \sim J_h \Rightarrow V_i(t_{h,j}) = V_j(t_{i,j}) \).

(ii) \( p < j \) and \( q \geq j \).

By symmetry, i.e. \( t_{i,e} = t_{h,e} \) where \( e \neq j, m-j+1, e = 1, 2, ..., j-1 \), we have \( I_{h,q}^i = I_{h,q}^1 = 0 \). The difference between \( t_{i,j} \) and \( t_{h,j} \) yields \( p < q \). Let us fix \( t_{i,j} = t_{h,j} \), then, \( p = q \). By assuming \( t_{i,m-j+1} < t_{h,m-j+1} \) the corresponding idle times on the last machine will be

\[ I_{h,q}^i = t_{i,q} + t_{i,m-j+1} - t_{h,m-j+1} - t_{h,m} \]

and

\[ I_{h,m}^i = t_{h,q} + t_{h,m-j+1} - t_{i,m-j+1} - t_{i,m} \]

Then, the difference of the makespan values is

\[ T_i - T_h = t_{i,q} + (t_{i,m-j+1} - t_{h,m-j+1}) - t_{h,q} + (t_{h,m-j+1} - t_{i,m-j+1}) \]
= t_{h,m-j+1} - t_{i,m-j+1} \quad \text{(4.36)}

Since \( t_{i,j} = t_{h,j} \),

Let without loss of generality \((h,i)\) be the optimal sequence, which means "\( J_h \) is preferred to \( J_i \)" or equivalently \( T_i > T_h \). Then, by (4.36) we have \( t_{h,m-j+1} > t_{i,m-j+1} \), which is consistent with the increasing assumption of \( V_{m-j+1}(\cdot) \) with respect to \( t_{i,m-j+1} \).

Then, by the corollary of Theorem 3.1, which indicates the optimality of the reverse problem with the reverse optimal sequence, we have "\( J_i \) is preferred to \( J_h \)". Notice that in the reverse problem \( t_{h,m-j+1} \) and \( t_{i,m-j+1} \) are related to the machine \( j \)

\[ J_i^R \text{ is preferred to } J_h^R \implies V_j(t_{h,m-j+1}) > V_j(t_{i,m-j+1}) \]

where \( t_{i,m-j+1} < t_{h,m-j+1} \). This indicates that \( V_j(\cdot) \) is an increasing function of the processing time values of \( M_j \).

(iii) \( q < j \) and \( p > j \)

This is similar to (ii), therefore the proof and the result are the same.

(iv) \( j < p \leq q \leq m - j \)

We consider the possibilities for \( p' \) and \( q' \), which are as follows
(a) \( j < p' < q' \)

Therefore

\[ I_{i . p'}^h = 0 \Rightarrow t_{i . l} + t_{i . j} - t_{h . j} - t_{h . p'} \leq 0 \]

and

\[ I_{h . p'}^i > 0 \Rightarrow t_{h . l} + t_{h . j} - t_{i . j} - t_{i . p'} > 0 \]

Thus,

\[ t_{i . j} < t_{h . j} \]

Let \( I_{i . p'}^h = 0 \) be the last vanishing idle time before \( q \), then, by Proposition 3.1

\[ t_{i . p'} - t_{h . p''} \leq 0 \]

where from \( I_{h . p''}^i > 0 \)

\[ t_{h . l} + t_{h . j} - t_{i . j} - t_{i . p''} > 0 \quad (4.37) \]

We can write for the idle time of \( M_q \) of the sequences as

\[ I_{i . q}^h = \max \{ 0 , \; t_{i . p''} - t_{h . q} \} \quad (4.38) \]

and

\[ I_{h . q}^i = 0 \Rightarrow t_{h . l} + t_{h . j} - t_{i . j} - t_{i . q} \leq 0 \quad (4.39) \]
By (4.37), (4.38) and (4.39), it results

\[ I^h_{i,q} = 0 \]

By fixing \( t_{h,m-j+1} = t_{i,m-j+1} \), since \( I^h_{i,q} = I^i_{h,q} = 0 \) and \( t_{i,e} = t_{h,e} \), \( e = q,...,m \), we will have

\[ I^h_{i,m} = I^i_{h,m} = 0. \]

By substituting the values of the processing and the idle times in the formula of the makespan

\[ T_h - T_i = t_{h,j} - t_{i,j} > 0 \]

\[ => T_h > T_i => J_i > J_h \]

\[ => U_j (t_{i,j}) > U_j (t_{h,j}) \text{ where } t_{h,j} > t_{i,j}. \]

(b) \( j < q' < p' \)

Therefore

\[ I^h_{i,q'} > 0 => t_{i,1} + t_{i,j} - t_{h,j} - t_{h,q'} > 0 \]

and

\[ I^i_{h,q'} = 0 => t_{h,1} + t_{h,j} - t_{i,j} - t_{i,q'} \leq 0 \]

Thus,
\[ t_{i,j} > t_{h,j} \]

Let \( I^i_{h,q''} = 0 \) be the last vanishing idle time before \( p' \), then, by Proposition 3.1, it holds

\[ I^i_{h,q''} = 0 \implies t_{h,q''} - t_{i,q''} \leq 0 \tag{4.40} \]

and

\[ I^h_{i,q''} > 0 \implies t_{h,1} + t_{h,j} - t_{i,j} - t_{i,q''} > 0 \tag{4.41} \]

We can write for the idle time of \( M_{p'} \) of the sequences as

\[ I^h_{i,p'} = 0 \max \{ 0, \ t_{i,q''} - t_{h,p} \} \tag{4.42} \]

and

\[ I^i_{h,p'} = 0 \implies t_{h,1} + t_{h,j} - t_{i,j} - t_{i,p'} \leq 0 \tag{4.43} \]

By (4.40) through (4.43)

\[ I^h_{i,p'} = 0 \]

We fix \( t_{h,m,j+1} = t_{i,m,j+1} \), since \( I^h_{i,q} = I^i_{h,q} = 0 \) and \( t_{i,e} = t_{h,e} \), where \( e = p', ..., m \), we will have

\[ I^h_{i,m} = I^i_{h,m} = 0. \]
By substituting the values of the processing and the idle times in the makespan formula

\[ T_i - T_h = t_{i,j} - t_{h,j} > 0 \]

\[ \Rightarrow T_i > T_h \Rightarrow J_h > J_i \]

\[ \Rightarrow V_j(t_{i,j}) < V_j(t_{h,j}) \text{ where } t_{h,j} < t_{i,j}. \]

(c) \( j < q' = p' \)

By fixing \( t_{i,j} = t_{h,j}, p = q \). The last machine idle times for the sequences are

\[ I_{i,m}^h = t_{i,q} + t_{i,m-j+1} - t_{h,m-j+1} - t_{h,m} > 0 \]

and

\[ I_{h,m}^i = t_{h,q} + t_{h,m-j+1} - t_{i,m-j+1} - t_{i,m} > 0 \]

Let

\[ t_{h,m-j+1} > t_{i,m-j+1} \]

Then

\[ T_i - T_h = t_{i,m-j+1} + (t_{h,m-j+1} - t_{i,m-j+1}) - t_{h,m-j+1} - (t_{i,m-j+1} - t_{h,m-j+1}) \]

\[ = t_{h,m-j+1} - t_{i,m-j+1}. \quad (4.44) \]
This case is similar to (ii)

(v) \( j < q \leq p \leq m - j + 1 \)

The proof and the final conclusion are similar to those of part (iv).

(vi) \( j < p \leq m - j + 1 < q \)

We consider the possible conditions of the first vanishing idle times after \( j \), which are as follow:

(a) \( j < p' < q' \)

Therefore

\[
I_{i,p'}^h = 0 \implies t_{i,1} + t_{i,j} - t_{h,j} - t_{h,p'} \leq 0
\]

and

\[
I_{h,p'}^i > 0 \implies t_{h,1} + t_{h,j} - t_{i,j} - t_{i,p'} > 0
\]

Thus,

\[
t_{i,j} < t_{h,j}
\]

Let \( t_{i,m-j+1} = t_{h,m-j+1} \), then, the case reduces to (iv)-a.

(b) \( j < q' < p' \)
Similarly to part (a), it results in $t_{h,i} < t_{i,i}$, then, by $t_{i,m-j+1} = t_{h,m-j+1}$ the case will be reduced to (iv)-b.

(c) $j < q' = p'$

Let $t_{i,m-j+1} = t_{h,m-j+1}$, the only feasible case will be $q = p$. Therefore, it holds that $I_{i,q}^h = 0$ and $I_{h,q}^l = 0$, which results in $I_{i,m}^h = 0$ and $I_{h,m}^l = 0$. By assuming $t_{h,i} < t_{i,j}$, it yields

$$T_i - T_h = t_{i,j} - t_{h,j} > 0$$

$$\Rightarrow T_i > T_h \Rightarrow J_h > J_i$$

$$\Rightarrow V_j(t_{i,j}) < V_j(t_{h,j}) \text{ where } t_{h,j} < t_{i,j}$$

(vii) $j < q \leq m - j + 1 < p$

The case is the same as (vi).

(viii) $m - j + 1 < p \leq q$

We consider the possible conditions of the first vanishing idle times after $j$, which are as follow:

(a) $j < p' < q'$
Similar to (iv)-(a)

\[ t_{h,j} < t_{h,j} \]

Let \( t_{i,m,j+1} = t_{h,m,j} \), then, since \( p \leq q \), then, similar to (iv)-(a), we conclude that \( l_{i,j}^h = 0 \) and \( l_{i,h}^i = 0 \), which results \( l_{h,m}^h = 0 \) and \( l_{h,m}^i = 0 \). Now since \( t_{i,j} < t_{h,j} \)

\[ T_h - T_i = t_{h,j} - t_{i,j} > 0 \]

\[ \Rightarrow T_h > T_i \quad \Rightarrow J_i > J_h \]

\[ \Rightarrow U_j (t_{h,j}) < U_j (t_{i,j}) \text{ where } t_{i,j} < t_{h,j} \]

(b) \( j < q < p' \)

Similar to (iv)-(b)

\[ t_{h,j} < t_{i,j} \]

Let \( t_{i,m,j+1} = t_{h,m,j} \), then, similar to (vi)-(a), since \( l_{i,j}^h = 0 \), hence \( l_{i,h}^i = 0 \). Therefore, \( l_{h,m}^h = l_{h,m}^i = 0 \). Now since \( t_{h,j} < t_{i,j} \)

\[ T_i - T_h = t_{i,j} - t_{h,j} > 0 \]

\[ \Rightarrow T_i > T_h \quad \Rightarrow J_h > J_i \]

\[ \Rightarrow V_j (t_{i,j}) < V_j (t_{h,j}) \text{ where } t_{h,j} < t_{i,j} \]
(c) \( j < q' = p' \)

Let \( t_{i, m-j+1} = t_{h, m-j+1} \), then, since \( t_{i, e} = t_{h, e} \), \( e = q'+1, \ldots, m \), we have \( I_{i, m}^h = I_{h, m}^i \). If \( t_{h, j} < t_{i, j} \), then the result is similar to part (b). \( \text{Q.E.D.} \)

Since the makespans of the problems are directly related to the values of the processing times, this indicates that it is plausible to take the parameter function \( f \), described in Section 4.2, as an identity function. We take \( V_j (\cdot) \), \( j = 1, 2, \ldots, m \), as linear functions of the processing times. Therefore, the unidimensional value functions for \( m \times n \) general makespan flowshop problems are introduced as

\[
V_j (t_{i, j}) = t^* - t_{i, j} \quad j = 1, 2, \ldots, \lfloor m/2 \rfloor
\]

and

\[
V_j (t_{i, j}) = t_{i, j} \quad j = \lfloor m/2 \rfloor + 1, \ldots, m
\]

In this section we derived the unidimensional value functions defined for machines of general makespan flowshop problems. The multiattribute approach requires to determine the MAV for the problem, which is done in the next section.
4.3.2 Assessment of the multiattribute value function

In this section the validity of the additivity assumption of the multiattribute value (MAV) function for an m×n general makespan flowshop problem is examined. The additivity assumption is proven to be valid for the case in which the unidimensional value functions are decreasing and decreasing for \( j \) and \( m-j+1 \), respectively, where \( j=1,\ldots,[m/2] \), \(([m/2] = m/2 \) when \( m \) is even; otherwise, it equals \( m/2-1 \)). However, there are makespan problems in which there exists \( j \) such that the unidimensional function is increasing and/or there exists \( m-j+1 \) such that the unidimensional function is decreasing. In these cases the additivity assumption has not proven to be valid. Nevertheless, the empirical results of Chapter 5 indicates that even if these assumptions are not strictly correct, an algorithm bases on an additive MAV still performs well is an excellent heuristic.

For this purpose the interface of MAV and scheduling theory is employed. Therefore, a background of MAV theory, which can be found in the MAV literature is introduced, see (Keeney and Raiffa, 1976; Dyer and Sarin, 1979; Chankong and Haimes, 1983).

Before introducing the relevant definitions it should be noted that each job \( J_i \) can be identified by its corresponding processing time values and the common technological order \( J_i = (t_{i,1}, \ldots, t_{i,m}), i=1,\ldots,n \). The precedence of the jobs are considered as MAV preference orders, i.e., for each pair of the jobs the
sequence \((J_i, J_k)\) indicates that \(J_i\) is preferred to \(J_k\). In fact it is assumed that any job \(J_i\) is considered as a point in an \(m\) dimensional space identified by \(t_{i,j}\) values, where \(j = 1, \ldots, m\).

Let consider \(J_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,m}) = (p_i, q_i)\), where \(p_i = (t_{i,1}, \ldots, t_{i,s})\), and \(q_i = (t_{i,s+1}, \ldots, t_{i,m})\), i.e., the attributes are partitioned into two sets shown by their different indices. Notice that each of the sets can have any combinations of the attribute levels.

**Definition 4.3** Given two jobs \(J_i = (t_{i,1}, t_{i,2}, \ldots, t_{i,m})\) and \(J_k = (t_{k,1}, t_{k,2}, \ldots, t_{k,m})\), let us fix \(q_i\) and \(q_k\) at \(q\), i.e., \(q_i = q_k = q\), where \(q_i = (t_{i,s+1}, \ldots, t_{i,m})\) and \(q_k = (t_{k,s+1}, \ldots, t_{k,m})\). \(p_i\) is *conditionally preferred to* \(p_k\), given \(q\), if and only if \((p_i, q)\) is *preferred to* \((p_k, q)\).

Let \(P\) stand for the corresponding attributes of the attributes values of \(p_i\) and \(p_k\). Similarly, attributes \(Q\), related to \(q_i = q_k = q\), include the complementary set of \(P\). Then, the following definition introduces the preferential relationship between \(P\) and \(Q\).

**Definition 4.4** The set of the attributes \(P\) is *preferentially independent of its complementary set\(Q\) if and only if the conditional preference of \(p_i\) to \(p_k\) for a given \(q\) does not depend on \(q\), i.e.,

\[
[(p_i, q) \text{ is preferred to } (p_j, q)] \Rightarrow [(p_i, q) \text{ is preferred to } (p_j, q)]
\]

for all \(q\).
Definition 4.5 Let us there exist attribute set $P = \{P_1, P_2, \ldots, P_m\}$. $P_1, P_2, \ldots, P_m$ are mutually preferentially independent if every subset $P'$ of $P$ be preferentially independent of its complementary set of the attributes.

So far we have introduced some definitions that provide preferential relations among the attributes. Based on these definitions, an important theorem has been introduced that determines the type of the MAV function, if it is additive. Although the following theorems are developed for utility function cases, as Dyer and Sarin (1979) state, they are also true for value functions by some substitution of language.

Theorem 4.1 Given a set of the attributes $a = (a_1, a_2, \ldots, a_m)$, with unidimensional value function $V_j(.)$ for the attribute $a_j$. There exists an additive MAV function for the attributes if and only if the attributes are mutually preferentially independent.

A formal proof of the theorem can be found in Debreu (1960), Fishburn (1970), and Krantz et al. (1971), as quoted in Keeney and Raiffa (1976).

According to Theorem 4.1, even for a modest number of the attributes a large number of preferential independence conditions should be verified. The following proposition efficiently reduces the number of the verifications.
Proposition 4.4 If every pair of attributes is preferentially independent of its complementary set, then, the attributes are mutually preferentially independent.

This proposition is a corollary to a theorem developed by Leontief (1947-(i), 1947-(ii)) and also proved in Gorman (1968-(i), 1968-(ii)). Based on Theorem 4.1 and Proposition 4.4, for the purpose of illustrating the additivity of MAV function for the general makespan flowshop problem it suffices to show that each pair of the attributes defined for the machines are preferentially independent from all other attributes. In the next section we show the validity of the MAV additivity assumption for the problem.

4.3.2.1 Additivity of MAV function

In this section we examine the additivity of MAV function. To do this task, first we determine the conditional preference for a pair of the attributes. Then, we consider the mutually preferential independence condition for the pair from their complementary set. Clearly, we should consider the above steps for all possible values of the attributes with respect to all possible idle time values of the machines. Notice that for a given set of the processing time values some sets of the idle times of the machines are impossible. From the scheduling aspect, this means that a feasibility study should be taken to determine all possible combinations of the
attribute values and their corresponding idle time values of the machines. For this purpose we require to use scheduling theory.

We remind that by the use of the transitivity property of MAV theory, we confine our search to an arbitrary pair of jobs. In fact MAV theory reduces the study of the \( m \times n \) general flowshop problem to an \( m \times 2 \) one. For this purpose, we fix the values of the other attributes for any arbitrary values. Let two attributes \( j \) and \( k, j < k \), be of concern. We select a pair of arbitrary jobs \( J_h \) and \( J_i \) such that

\[
t_{h,i} = t_{i,j} = t_i \quad \text{where } l \neq j,k
\]

(4.45)

Two sequences can be constructed with the jobs, which are \((h,i)\) and \((i,h)\), where \((h,i)\) indicates that \( J_h \) precedes \( J_i \). Let the makespans and the idle times of the sequences \((h,i)\) and \((i,h)\) be denoted by \( T_h \) and \( I^h \), \( T_i \) and \( I^i \), respectively. (Notice that \( I_{h,j}^i \) indicates the idle time of the job \( J_j \) on the machine \( M_j \) for the sequence \((i,h))\). The makespan values for the sequences are

\[
T_h = \sum_{l \neq j,k} (t_{h,i} + t_{h,k} + I^h_{i,m} + t_{h,m} + t_{i,m}) \quad \text{where } l \neq j,k
\]

(4.46)

and

\[
T_i = \sum_{l \neq j,k} (t_{i,j} + t_{i,k} + I^i_{h,m} + t_{i,m} + t_{h,m}) \quad \text{where } l \neq j,k
\]

(4.47)

By (4.45), (4.46) and (4.47)
\[ T_i - T_j = (t_{i,j} + t_{i,k} + I_{h,m}^i) - (t_{h,j} + t_{h,k} + I_{i,m}^h) \] (4.48)

Without loss of generality, let \((h,i)\) be the optimal sequence for a certain level of \(t_{h,j} = t_{i,j} = t_i\) where \(i \neq j, k\). This indicates that in terms of multiattribute alternative ranking \(J_h\) is preferred to \(J_i\), which results in \(T_h < T_i\). Then, by (4.48)

\[ (t_{i,j} + t_{i,k} + I_{h,m}^i) - (t_{h,j} + t_{h,k} + I_{i,m}^h) > 0 \] (4.49)

By scheduling theory, discussed in Chapter 3, different processing times, or equivalently the attribute levels, may affect the idle time values \(I^h\) and \(I^i\). We should consider the conditional preference of the pair of the attributes for the unidimensional value functions, determined in the last section. Therefore, there exist three possible situations for the pair of the attributes with respect to the unidimensional value functions, which are

(i) \(k, j = 1, 2, ..., \lceil m/2 \rceil\),

(ii) \(k = 1, 2, ..., \lceil m/2 \rceil ; j = m - \lfloor m/2 \rfloor + 1, ..., m\),

(iii) \(k, j = m - \lfloor m/2 \rfloor + 1, ..., m\),

where \(\lceil m/2 \rceil = m/2\) if \(m\) is even; otherwise, \(\lfloor m/2 \rfloor = m/2 - 1\).

As cited before, at first, we determine the conditional preference of the pair of the attributes. Before dealing with that we introduce the following proposition indicating that it is sufficient to consider either case (i) or case (iii).
Proposition 4.5 Given an $m \times n$ general makespan flowshop problem and a pair of attributes $j$ and $k$. The attributes $j$ and $k$ are mutually independent from their complementary set, if and only if the attributes $m-j+1$ and $m-k+1$ are mutually preferentially independent from their complementary set.

Proof: For necessary condition, let the attributes $j$ and $k$ be mutually preferentially independent from their complementary set. That is, by definition, for any possible values of the pair of the attributes, they are preferentially independent of the other attribute levels. We reverse both the problem and the sequence. For the reverse problem with the reverse sequence, the levels of the attributes $m-j+1$ and $m-k+1$ are the same. Therefore, by Theorem 3.1, the preferential relationship determined by the makespan values are preserved for the reverse problem. Thus, we can conclude that for any possible values of the pair of the attributes $m-j+1$ and $m-k+1$ are mutually preferentially independent from their complementary set.

For the sufficiency, we use Remark 3.1, which indicates that each reverse problem can be considered as an original one. Now let the attributes $m-j+1$ and $m-k+1$ be mutually preferentially independent. Then, we consider the reverse problem as the new original and repeat the argument given in the necessary part for the new original problem. Q.E.D.
By Proposition 4.4 we confine our study to the conditional preference of any pair of the attributes to the cases (i) and (ii). For the conditional preference we choose the specified values as
\[ t_{h,w} = t_{i,w} = t_{w} = 0 \quad \text{where} \quad w \neq j, k, i = 1, 2, ..., m. \]  
(4.50)

Now, we consider case (i) and search for all possible values of the levels of the attribute \( j \) and \( k \). Therefore, we consider all possibilities of the idle times in (4.49), which indicates all possible values of \( t_{yj} = t_{yk} \), \( y \neq h, i \). Notice that (4.49) has been obtained by assuming \( T_h < T_i \). To deal with (4.49) we consider two subcases, which are,
\[ t_{i,j} + t_{i,k} > t_{h,j} + t_{h,k} \]  
(4.51)
\[ t_{i,j} + t_{i,k} \leq t_{h,j} + t_{h,k} \]  
(4.52)

Since the idle times can take zero and positive values, there exist four possible subcases for (I), as follows:

(I)\(_1^1\): \( I_{h,m}^h > 0 \) and \( I_{h,m}^i > 0 \)

\[ I_{i,m}^h > 0 \Rightarrow t_{i,j} - t_{h,k} > 0 \Rightarrow t_{i,j} > t_{h,k} \]

and

\[ I_{h,m}^i > 0 \Rightarrow t_{h,j} - t_{i,k} > 0 \Rightarrow t_{h,j} > t_{i,k} \]

By substituting the values of the idle times in (4.49),
\[( t_{i,j} + t_{i,k} - t_{h,j} + t_{i,k} ) - ( t_{h,j} + t_{h,k} - t_{i,j} + t_{h,k} ) > 0 \]  \tag{4.53}

\[\Rightarrow t_{i,k} < t_{i,j} \Rightarrow \text{contradiction} \Rightarrow \text{the case is infeasible.}\]

(I)\textsubscript{2}: \( I_{l,m}^h = 0 \) and \( I_{l,m}^i > 0 \)

\[ I_{l,m}^h = 0 \Rightarrow t_{i,j} - t_{h,k} \leq 0 \Rightarrow t_{i,j} \leq t_{h,k} . \] \tag{4.54}

and

\[ I_{l,m}^i > 0 \Rightarrow t_{h,j} - t_{i,k} > 0 \Rightarrow t_{h,j} > t_{i,k} \]

By substituting the values of the idle times in (4.49),

\[ ( t_{i,j} + t_{i,k} - t_{h,j} + t_{i,k} ) - ( t_{h,j} + t_{h,k} ) > 0 \] \tag{4.55}

\[\Rightarrow t_{i,k} < t_{i,j} \Rightarrow \text{contradicts (4.54)} \Rightarrow \text{the case is infeasible.}\]

(I)\textsubscript{3}: \( I_{l,m}^h > 0 \) and \( I_{l,m}^i = 0 \)

\[ I_{l,m}^h > 0 \Rightarrow t_{i,j} - t_{h,k} > 0 \Rightarrow t_{i,j} > t_{h,k} . \] \tag{4.56}

and

\[ I_{l,m}^i = 0 \Rightarrow t_{h,j} - t_{i,k} \leq 0 \Rightarrow t_{h,j} \leq t_{i,k} \]

By substituting the values of the idle times in (4.49)

\[ ( t_{i,j} + t_{i,k} ) - ( t_{h,j} + t_{h,k} - t_{i,j} + t_{h,k} ) > 0 \] \tag{4.57}

\[\Rightarrow t_{i,k} > t_{h,j} . \] \tag{4.58}
By (4.58) and (4.51), it results that
\[ t_{i,j} > t_{h,k} \text{ and } t_{h,j} < t_{i,k} \text{.} \]  \hfill (4.59)

(I): \quad I^h_{i,m} = 0 \text{ and } I^i_{h,m} = 0

\[ I^h_{i,m} = 0 \implies t_{i,j} - t_{h,k} \leq 0 \implies t_{i,j} \leq t_{h,k} \text{.} \]  \hfill (4.60)

\[ I^i_{h,m} = 0 \implies t_{h,j} - t_{i,k} \leq 0 \implies t_{h,j} \leq t_{i,k} \]  \hfill (4.61)

By substituting the values of the idle times in (4.49),
\[ (t_{i,j} + t_{i,k}) - (t_{h,j} + t_{h,k}) > 0 \]  \hfill (4.62)

(II): \quad For the subcase (II), we rewrite (4.51)

\[ t_{i,j} + t_{i,k} \leq t_{h,j} + t_{h,k} \]  \hfill (4.52)

By substituting (4.52) in (4.59),
\[ I^h_{i,m} - I^i_{h,m} < 0 \implies 0 \leq I^h_{i,m} - I^i_{h,m} \]

Thus, two possible cases exist, which are

(II): \quad I^h_{i,m} > 0 \text{ and } I^i_{h,m} > 0

\[ I^h_{i,m} > 0 \implies t_{i,j} - t_{h,k} > 0 \implies t_{i,j} > t_{h,k} \]

and
I^1_{h,m} > 0 \Rightarrow t_{h,j} - t_{i,k} > 0 \Rightarrow t_{h,j} > t_{i,k}

By substituting the values of the idle times in (4.49), finally

\[
( t_{i,j} + t_{i,k} + t_{i,j} + t_{h,k} ) - ( t_{h,j} + t_{h,k} + t_{h,j} + t_{i,k} ) > 0
\]

(4.63)

\Rightarrow t_{i,k} < t_{i,k} \Rightarrow \text{contradiction} \Rightarrow \text{the case is infeasible.}

(II)\_2 : I^h_{i,m} = 0 \text{ and } I^1_{h,m} > 0

I^h_{i,m} = 0 \Rightarrow t_{i,j} - t_{h,k} \leq 0 \Rightarrow t_{i,j} \leq t_{h,k}

(4.64)

and

I^1_{h,m} > 0 \Rightarrow t_{h,j} - t_{i,k} > 0 \Rightarrow t_{h,j} > t_{i,k}

By substituting the values of the idle times in (4.49), finally,

\[
t_{i,j} > t_{h,k} \Rightarrow \text{contradiction with (4.64)} \Rightarrow \text{the case is infeasible.}
\]

The consideration of all of the possible cases are being done through the analysis of all idle time combinations of the original and reverse problems, and each of the idle times can either zero or positive values. Therefore, for \( m \) attributes there exist \( 4^{m-1} \) combinations of zero and positive idle time values. By taking advantage of \( t_{h,w} = t_{i,w} = t_w \), where \( w \in \{ j, k \} \), \( i=1,2,...,m \), the number of the possible combinations will shrink. This point is considered in the following lemma.
Lemma 4.1 Given an \( m \times n \) general makespan with a given sequence \((h,i)\) and known \( t_{h,j}, t_{h,k}, t_{i,j}, \) and \( t_{i,k}. \) Let \( t_{h,1} = t_{i,1} = t_1 \geq 0, l = a+1, ..., a+g-1, \) where \( a+1 > j \) and \( a+g < k. \) If \( \mathbf{l}_{h,i,1} > 0, l = a+1, ..., a+g, \) where \( a+1 > j \) and \( a+g \leq k, \) then, replacing \( t_1 \) by zero will not affect the value of \( \mathbf{l}_{h,i,a+g}. \)

**Proof:** We write the formula for \( \mathbf{l}_{h,i,1} > 0, l = a+1, ..., a+g-1, \) where \( a+1 > j \) and \( a+g < k, \) which are

\[
\mathbf{l}_{h,i,a+1} = \max \{ 0, \mathbf{l}_{h,i,a} + t_{i,a} - t_{h,a+1} \} > 0
\]

\[
\mathbf{l}_{h,i,a+2} = \max \{ 0, \mathbf{l}_{h,i,a+1} + t_{i,a+1} - t_{h,a+2} \} > 0
\]

\[
\vdots
\]

\[
\mathbf{l}_{h,i,a+1} = \max \{ 0, \mathbf{l}_{h,i,a+g} + t_{i,a+g} - t_{h,a+g} \} > 0
\]

By summing up both sides of the equations for the idle times, and considering \( t_{h,1} = t_{i,1} = t_1 \geq 0, l = a+1, ..., a+g-1, \) where \( a+1 > j \) and \( a+g \leq k, \)

\[
\mathbf{l}_{h,i,a+1} = \mathbf{l}_{h,i,a} + t_{i,a} - t_{h,a+g}
\]

(4.65)

Clearly, (4.65) can be achieved if \( t_{h,1} = t_{i,1} = t_1 = 0, l = a+1, ..., a+g-1, \)

where \( a+1 > j \) and \( a+g \leq k \) hold. Q.E.D.

In fact by Lemma 4.1 and the use of Proposition 3.1 we can confine our study to the last vanishing idle times before and after machines \( M_j \) and \( M_k, \) where their corresponding attributes are of concern. That is, instead of
studying problem (P1) we can analyze problem (P2), as their corresponding processing time matrices are as follows.

\[
\text{(P1)} \begin{bmatrix}
t_{h1} & \cdots & t_{hj} & \cdots & t_{hk} & \cdots & t_{hm} \\
t_{i1} & \cdots & t_{ij} & \cdots & t_{ik} & \cdots & t_{im}
\end{bmatrix}
\]

and

\[
\text{(P2)} \begin{bmatrix}
t_{he} & \cdots & t_{hj} & \cdots & t_{hp} & \cdots & t_{hs} \\
t_{ie} & \cdots & t_{ij} & \cdots & t_{ip} & \cdots & t_{is}
\end{bmatrix}
\]

Notice that the indices e, p, and s represent the machines $M_e$, $M_p$, and $M_s$ whose idle times are possibly the last vanishing in the intervals $j=1,...,j-1$, and $j = j+1,...,p-1$, and $j = k+1,...,m$, respectively.

Now we continue to examine the validity of (4.49) for all possible values of the attributes $t_{h,1} = t_{h,1}$, $l = e, p, s$. Notice that we already have (i) determined the feasible space of $t_{h,1} = t_{h,1}$, $t_{h,j}$, $t_{h,k}$, and $t_{i,k}$ for part (i), obtained from (I) and (I)$. Therefore, the feasible space has two parts. The first part, say (A), determined by (4.49),(4.56) and(4.57), is

\[
\begin{align*}
t_{ik} > t_{h,j} & \quad \text{and} \quad t_{i,j} > t_{h,k} \\
t_{i,j} + t_{i,k} > t_{h,j} + t_{h,k}
\end{align*}
\]

The second part, say (B), determined by (4.49),(4.60) and(4.61), is
Operationally speaking, for the study of all combinations of the cases we need to split both (A) and (B) by introducing more cases, i.e.,
either
1. \( t_{h,k} > t_{i,k} \)

or
2. \( t_{h,k} \leq t_{i,k} \)

Therefore, for part (i) there exist four subspaces, and each of them have to be examined for the validity of (4.49). Also, for each of the subcases there exists \( (4)^4 \) combinatorial cases (notice that the possible cases for \( I_{h,i,e} \) and \( I_{h,e} \) can be considered by their effects on \( I_{h,i,j} \) and \( I_{h,j} \) as shown in Appendix 2). Therefore, \( (4)(4)^4 = 1024 \) combinations are required to be searched for part (i). We remind that by Proposition 4.4, the conclusion for part (i) is also valid for part (iii).

Let (A) with constraints (1) and (2) be denoted by \( (A)_1 \) and \( (A)_2 \), respectively, and similar notation be taken for (B). We confine our study to \( (A)_1 \) and \( (B)_1 \), which are determined by the following sets of constraints.

\[
\begin{align*}
& t_{i,k} > t_{h,j} \quad \text{and} \quad t_{i,j} > t_{h,k} \\
& t_{h,k} > t_{i,k} \\
& t_{i,j} + t_{i,k} > t_{h,j} + t_{h,k}
\end{align*}
\]

and
We examine 512 cases. Each case examination requires need two steps, which are feasibility study and testing the case for the validity of (4.49). Appendix 2 contains a detailed study of \((A)_{1}\).

The result of examining 512 cases indicates that 379 cases are infeasible. Regardless of the attribute values, (4.49) holds for 111 cases out of 133 feasible ones. That is, the pair of attributes \(j\) and \(k\) is mutually preferentially independent from its complementary set for 111 cases. For 19 cases, (4.49) holds conditionally, i.e. the conditional preference of the attributes holds. Finally, there exist three cases for those (4.49) is violated, which are the worse cases provided by MAV approach. Comparative analysis of these cases illustrate that a set of assumptions can be developed to make the validity of the attributes "mutually preferential independence" plausible. For instance, if an upper bound for processing times be assigned and/or the distribution of the processing times be taken as random, which are acceptable assumptions in the real world problems, it can be shown that the occurrence of the worse cases is less likely. Although in the next chapter it will be shown the proposed heuristic method based on the MAV approach, without any further modifying assumptions, is more powerful than the existing efficient heuristics, specially for problems with very biased, irregular,
and unbalanced processing times. A detailed analysis of the assumptions will be provided later.

In brief, in this section we have shown that it is heuristically plausible that the MAV function for the general makespan flowshop problem are additive function. That is, the MAV for job \( i \) can be stated as

\[
V(t_{i,1}, ..., t_{i,m}) = \sum_{j=1}^{m} w_j V_j(t_{i,j})
\]

where \( w_j \), \( j=1,...,m \), which are the scaling weights.

For the assessment of \( V(t_{i,1}, ..., t_{i,m}) \) we need to determine both \( V_j(t_{i,1}, ..., t_{i,m}) \) and the scaling weights \( w_j \), \( j=1,...,n \), which are derived in the next section.

4.3.2.2 Scaling weights of the MAV function

The additive scaling weights \( w_j \), \( j=1,...,n \), indicate the relative significance of the unidimensional value functions in the MAV function. In terms of scheduling theory, the same processing time on different machines may have different impact on the rank of a certain job to obtain optimal permutation schedule.

In this section we illustrate that the scaling weights of the MAV function for the general flowshop problem are symmetric with respect to the imaginary axis dividing the processing time matrix into two parts with
equal number of machines. That is, the scaling weights of the machines $M_j$ and $M_{m-j+1}$ where $j = 1, ..., [m/2]$, as $[m/2]$ given before, are the same.

**Proposition 4.6** Given an $m \times n$ general makespan flowshop problem, if the MAV function be additive, to obtain an optimal sequence the nonnegative scaling weights of the unidimensional value functions $V_i(\cdot)$ and $V_{m-j+1}(\cdot)$ are the same.

**Proof:** By contradiction. Let the scaling weights of the unidimensional value functions $V_i(\cdot)$ and $V_{m-j+1}(\cdot)$ be different. Choose two different jobs $J_h$ and $J_i$ such that

$$t_{h,j} = t_{i,j} \quad \text{where} \quad j \neq k, m-k+1, j=1,2,...,m. \quad (4.65)$$

Without loss of generality, suppose the sequence $(h,i)$, which indicates $J_h$ precedes $J_i$, be the optimal one. Then, by the unidimensional value functions derived in the last section, we have for $(h,i)$,

$$w_j(t^{**-} - t_{h,j}) + w_{m-j+1}(t_{h,m-j+1}) > w_j(t^{**} - t_{i,j}) + w_{m-j+1}(t_{i,m-j+1}) \quad (4.66)$$

where $t^{**} = \max t_{k,1}, \quad k=1,...,n; \quad l=1,...,m$.

By the corollary of Theorem 3.1 we also have for the reverse problem with the reverse sequence

$$w_j(t^{**} - t_{i,m-j+1}) + w_{m-j+1}(t_{i,m-j+1}) > w_j(t^{**} - t_{h,m-j+1}) + w_{m-j+1}(t_{h,j}) \quad (4.67)$$
where \( t^{**} = \max t_{k,1}, \quad k = 1, \ldots, n; \quad l = 1, \ldots, m. \)

We rewrite (4.66) and (4.67) as

\[
w_j \left( t_{i,j} - t_{h,j} \right) > w_{m-j+1} \left( t_{i,m-j+1} - t_{h,m-j+1} \right)
\]

(4.68)

and

\[
w_{m-j+1} \left( t_{i,j} - t_{h,j} \right) > w_j \left( t_{i,m-j+1} - t_{h,m-j+1} \right)
\]

(4.69)

Since \( w_j \) is different from \( w_{m-j+1} \), let \( w_j > w_{m-j+1} \), it is sufficient to show that (4.68) and (4.69) do not hold simultaneously. For instance, if

\[
( t_{i,j} - t_{h,j} ) > ( t_{i,m-j+1} - t_{h,m-j+1} ) > 0
\]

hold, then (4.68) holds, but (4.69) does not. The same argument is true for the case when \( w_j > w_{m-j+1} \) is supposed. Thus,

\[
w_j = w_{m-j+1}, \quad j = 1, \ldots, [m/2].
\]

Q.E.D.

In the next proposition we try to find some relationship between the scaling weights. Notice that by the use of Proposition 4.6 we can confine our study to the \( M_j, \ j = 1, \ldots, [m/2] \), where \([m/2] = m/2\), if \( m \) is even; otherwise \([m/2] = m/2 - 1\).
Proposition 4.7 For an $m \times n$ general makespan flowshop problem, if MAV function is additive, then, for nonnegative scaling weights, it holds that

$$\frac{w_j}{w_k} > 1 \quad \text{where} \quad j < k ; \ j, k = 1, \ldots, [m/2], \ \text{and} \ [m/2] = m/2,$$

if $m$ is even; otherwise, $[m/2] = m/2 - 1$.

Note that the following relation satisfies the relationship $w_j / w_k > 1$:

$$w_j / w_k = (m-j / m-k)^\alpha \text{ where } \alpha > 0$$

Proof: Let us have two different jobs such that

$$t_{h,k} = t_{i,k} \quad \text{where} \quad k \neq j, j+1 ; \ j=1, 2, \ldots, [m/2] - 1. \quad (4.70)$$

Let assume the sequence $(h,i)$ provide be the optimal one, then by considering the corresponding $V_j(.)$ and $V_{j+1}(.)$, where $j=1, 2, \ldots, [m/2] - 1$,

$$w_j(t^{**}_{i, j} + w_{j+1}(t^{**}_{h, j+1}) > w_j(t^{**}_{i, j} + w_{j+1}(t^{**}_{i, j+1}) \quad (4.71)$$

which results in

$$w_j (t_{i, j} - t_{h, j}) > w_{j+1} (t_{h, j+1} - t_{i, j+1}) \quad (4.72)$$

(4.72) holds for all cases if only $w_j > w_{j+1}$ holds. We also can write for $j > k$
\[
\left( \frac{w_j}{w_k} \right) = \left( \frac{w_j}{w_{j+1}} \right) \left( \frac{w_{j+1}}{w_{j+2}} \right) \ldots \left( \frac{w_{k-1}}{w_k} \right) > 1 \quad (4.73)
\]

In fact as the machine number increases its corresponding weight decreases. To provide a relationship between the weights \( w_j \) and \( w_k \) in (4.73), we relate the weights to their corresponding machine numbers. Hence, for \( w_j \) and \( w_k \), we can write

\[
w_j > w_k \implies \left( \frac{w_j}{w_k} \right) = \left[ \frac{(m/2-j)}{(m/2-k)} \right]^\alpha > 1 \quad \text{where } \alpha > 0.
\]

Q.E.D.

From Proposition 4.7 we can see that \( w_j \) is proportional to \( (m/2-j)^\alpha \), where \( \alpha \) is an unknown parameter. Since the scaling weight is of concern we heuristically consider \( w_j = (m/2-j)^\alpha \). Then, by Proposition 4.6, we can conclude that \( w_j = w_{m-j+1} = (m/2-j+1)^\alpha \). In fact by the use of Propositions 4.6 and 4.7 the \( m \) unknown scaling weights have been reduced to one parameter \( \alpha > 0 \), which will be introduced in detail in the next chapter.

Finally, the MAV function derived for an \( m \times n \) general makespan flowshop problem

\[
V(t_{i,1}, \ldots, t_{i,m}) = \sum_{j=0}^{m/2} \left( \frac{m}{2-j} \right)^\alpha (V_{j+1}(t_{i,j+1}) + V_{m-j}(t_{i,m-j}))
\]

where \( \alpha > 0 \), \( V_{j+1}(t_{i,j}) = t^{**} - t_{i,j+1} \), and \( V_{m-j}(t_{i,m-j}) = t_{i,j} \), \( j=1, \ldots, [m/2] \).
The algorithm is developed and the computational results are given in the next chapter.

4.4 Summary and conclusions

4.4.1 Summary

In this chapter a multiattribute formulation for general flowshop problems is introduced. The formulation has been applied to the problems having makespan as the performance objective function. By the use of scheduling theory the qualitative characteristics of the unidimensional value functions for the machines are determined. To afford this task, in Propositions 4.1 and 4.2 the functions for the first and the last machines are obtained. Then, in Proposition 4.3 the characteristic of the value functions for each symmetric machines $M_j$ and $M_{m-j+1}$. $j=1,...,\lfloor m/2\rfloor$ are determined. Then, the additivity of the multiattribute value function has been examined and discussed in detail.

4.4.2 Conclusions

In this chapter we have determined the unidimensional value functions for the machines $M_j$ in a general flowshop problem, which are decreasing, given the unidimensional value functions for $M_{m-j+1}$ be increasing. The optimal sequential schedules indicate that the additivity assumption for the multiattribute is heuristically plausible to obtain a near optimal solution. The characteristics of the unidimensional value functions and the additivity assumption conclude that the
scaling weights for the machines $M_j$ and $M_{m-j+1}$ are the same. Then, we
determined a relation between the scaling weights and the indices of the machines
in the processing vector. Finally, an additive MAV has been derived for the
problem.
CHAPTER 5

NEW ALGORITHM AND COMPUTATIONAL RESULTS

5.1 Introduction

The theoretical background of the MAV approach to general makespan flowshop problem is provided in the last chapter. In this chapter a new functional heuristic algorithm is constructed and presented in more detail. The algorithm is compared to some well-known existing procedures, being used in the flowshops (Karimi and Ku, 1988). As cited before there exist several studies compared the methods, and have stated the superiority of CDS method to the others from various aspects, for example, see Setiapura (1980).

5.2 A new heuristic

Based on the theoretical achievements of the last chapter we can introduce a heuristic algorithm for an m×n general makespan flowshop problem. The algorithm has the following steps, note that \( \alpha > 0 \) will given in the next section.
Step 1. Determine the MAV function value, \( V(t_{i_1}, \ldots, t_{i_n}) \), for job \( J_i \), where 
\[
V(t_{i_1}, \ldots, t_{i_m}) = \sum_{j=0}^{m/2} (m/2-j) \alpha (V_{j+1}(t_{i_{j+1}}) + V_{m-j}(t_{i_{m-j}}))
\]
where \( \alpha > 0 \), and 
\[
V_{j+1}(t_{i_{j+1}}) = t^{**} - t_{i_{j+1}} \), and 
and 
\[
V_{m-j}(t_{i_{m-j}}) = t_{i_{m-j}} \), \( j=0, \ldots, [m/2] \).
\]
Notice: Since the relative value of the MAV function for each job does not depend on the value of \( t^{**} \), we can take \( t^{**} \) as zero.

Step 2. Sort the MAV function value, \( V(t_{i_1}, \ldots, t_{i_m}) \), for job \( J_i \), \( i = 1, \ldots, n \), in decreasing order for each value of the parameter \( \alpha \). The order of the jobs provides a sequence corresponding to the values of the parameter \( \alpha \).

Step 3. Determine the makespan value for the sequences obtained for each value of \( \alpha \).

Step 4. Sort the makespan values, and determine the minimum one(s). The corresponding sequence is taken as the final sequence.
5.3 Computational results

The computational results are presented in two parts. The first part focuses on the unknown parameter $\alpha$, which is a key factor for applying the algorithm. In the second part the algorithm is compared to some other well-known efficient algorithms.

5.3.1 Evaluating parameter

The parameter $\alpha$ has been evaluated experimentally. For this purpose more than 700 problems of different machine and job numbers have been generated, where $m= 5, 15, 30, 45, 60, 75, 90$ and $n= 5, 15, 30, 40, 50, 60, 75, 90$. The processing times are randomly generated using a uniform distribution between 0 and 100. The problems are solved for 150 equidistant values of $\alpha \in (0,15)$, i.e. the step size for the values of $\alpha$ in the interval is 0.1. Then, the optimal $\alpha$ values, which provide the least makespan, have been taken. A bar chart of the distribution of the parameter $\alpha$ is shown in Figure 5.1.

According to Figure 5.1, the frequency of optimal $\alpha$ values is decreasing in the interval (0,15). For instance, the chance of optimal $\alpha$ value to be in (0,1), (1,2), or (2,3) is 34.0, 25.9, and 20.1 percent, respectively. Finally, 90.3 and 95.9 percent of the optimal are in intervals
(0,4) and (0,5), respectively. Therefore, it is plausible to consider (0,5) as a proper range for optimal $\alpha$ values.\footnote{Note that if the optimal $\alpha$ is very high (say, greater than 5), then, the method in effect reduces to that of Page (1961), in which only the first and the last machines are considered.}

The next point regarding the parameter $\alpha$ is selecting a proper step size within interval (0,5). The number of the step sizes may affect the performance of the algorithm, which will be discussed in detail in the next section.

5.3.2 Comparison of algorithms

The proposed algorithm has been compared with four other algorithms, which are introduced in the literature review and applied in the shops, (Setiapura 1980; Karimi and Ku, 1988; Windmer and Hertz, 1989). The algorithms are Palmer(1965), Gupta (1969), RA of Dannenberg (1977), and CDS(1977) heuristics. Moreover, the performance of the new algorithm has been compared to one of the semi-exhaustive algorithm NEH (Newaz et al.,1983). Three criteria for the comparison of the algorithms are considered, which are makespan value, CPU time, and the number of better solutions provided.

For a precise analysis of the performance of the algorithms, 2930 flowshop problems are produced, where the processing times are generated randomly from a uniform distribution in [0, 100]. The problems are grouped according to the size of the shops, $m$, and the number of the jobs, $n$,
Figure 5.1

The bar-chart distribution of the optimal alpha value versus alpha values.
to be scheduled. The size of the shops are determined by the number of their machines, which are grouped as small, medium, and large for up to 20, between 20 and 50, and more than 50 machines, respectively. Similar to the shop sizes, the same group sizing has been taking for the number of jobs. Hence, there exist 9 types of flowshop problems, i.e. small shop with small, medium, and large number of jobs, and similarly for medium and large shops. In our study we compare 3 types of the flowshop problems. For each group of the problems \( n \) and \( m \) are increased using step sizes 1, 5 and 5 for Types 1, 2 and 3, respectively. The types of the problems are defined as follows.

Type 1 problems contain those flowshop problems having small shops and a small number of the jobs to be processed. Type 2 includes the problems with medium shops and a medium number of the jobs. Finally, Type 3 encompasses the problems having a large number of the jobs to be processed in the large shops.

Unfortunately, most of the comparative studies of the flowshop algorithm are done for up to Type 2 problems. The main point for Type 1 problems is for most algorithms, the CPU times is at most one minute on an IBM-AT compatible personal computers with math co-processor which does not show the complexity of the algorithms for the problems of Types 3, which needs much more computational effort.

Two other points should be mentioned in the comparative study of the algorithms. First, the generality of the flowshop problems indicates the probability that a job does not need a certain machine. The other is the unbalanced load, which indicates the probability that the processing times of a job be very large, e.g. ten times larger, than the average processing time in the
problem. The reason d'etre of these concepts, as discussed in the literature review, is the performance of the functional algorithms worsen when the problems have either high generality or unbalanced load or both. The algorithms are also studied for the cases having different generality levels and unbalanced loads.

**Small (Type 1) problems**

In this part we consider the small problems (Type 1). The results for the makespan values of the algorithms, in percentage, comparing to the makespan values of the MAV algorithm have been shown in Figure 5.2. The MAV algorithm has been categorized, according to the number of the step sizes taken in the interval (0,5), e.g., MAV 10 indicates that the number of the equal step sizes in the interval is 10.
Figure 5.2

The percentage of the makespan values of the algorithms versus generality values comparing to MAV50 for small (Type 1) problems.

Notice: Generality indicates the probability a job that skip a machine.
As shown the MAV algorithm provides better makespan values comparing to all other algorithms. Also, as the generality of the problems increases the relative performance of MAV is much better than of the other algorithms. In fact using MAV provides a saving of 4.82 percent of shop time comparing to CDS. As the number of the step sizes of MAV increases the relative performance of the algorithm also improves.

Another criterion for the comparison of the algorithms is the number of better solutions provided by each algorithm comparing to MAV, which is shown in Table 5.1. Based on the results MAV provides 81.6 percent of times better or equal than CDS for small (Type 1) problems and more than 95 and almost 100 percent of times better than or equal to Gupta and Palmer methods.

The other criterion for the evaluation of the algorithms is the required computational effort. In Figure 5.3 the relative CPU times for the algorithms, comparing to MAV, are shown. Note that the computational efforts of most algorithms are not significant, i.e. about one second in average, for small (Type 1) problems.

As shown the computational effort almost does not depends on the generality levels. Although the CPU time of MAV50 is more than the one of CDS the actual average difference is 1.43 second and maximum difference is 18.89 seconds on an IBM-AT compatible personal computer with math. co-processor. For the purpose of comparing on the basis of the same computational effort MAV 10 also has been shown, where its superiority to CDS has already been shown in Figure 5.2.
Table 5.1
The number of better makespan values of the algorithms comparing to MAV50 for small (Type 1) problems with different generality values

<table>
<thead>
<tr>
<th>MAV50</th>
<th>Generality=0</th>
<th>Generality=0.3</th>
<th>Generality=0.6</th>
<th>Generality=0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>more</td>
<td>less</td>
<td>tie</td>
<td>more</td>
</tr>
<tr>
<td>Dan. RA</td>
<td>198</td>
<td>47</td>
<td>11</td>
<td>191</td>
</tr>
<tr>
<td>Gupta</td>
<td>238</td>
<td>13</td>
<td>5</td>
<td>252</td>
</tr>
<tr>
<td>CDS</td>
<td>198</td>
<td>47</td>
<td>11</td>
<td>189</td>
</tr>
<tr>
<td>Palmer</td>
<td>217</td>
<td>0</td>
<td>39</td>
<td>212</td>
</tr>
</tbody>
</table>

The total number of the problems for each generality level = 256.
The sizes of the problems: \( n = 5 \) to \( 20 \) step 1; \( m = 5 \) to \( 20 \) step 1.
more: the number of the schedules provided worse than MAV.
less: the number of the schedules provided better than MAV.
tie: the number of the schedules provided similar to MAV.
Notice: Generality indicates the probability that a job skip a machine.
Figure 5.3

The percentage of the running time values of the algorithms versus generality values comparing to MAV50 for small (Type 1) problems.

Notice: Generality indicates the probability a job that skip a machine.
Medium (Type 2) problems

As cited before medium (Type2) problems include the problems having a medium number of the jobs to be processed in a medium shop. The results for the makespan values of the algorithms, in percentage, comparing to the makespan values of the MAV algorithm have been shown in Figure 5.4. As shown, the MAV algorithm provides smaller makespan values with respect to all other algorithms. Also, as the generality of the problems increases the relative performance of MAV is much better than of the other algorithms. In fact using MAV provides a saving of 4.35, 6.74, and 6.31 percent of shop time comparing to CDS for generality levels 0.3, 0.6 and 0.9.

Also, the performances of MAV20 has been shown, which is superior to all other algorithms but inferior to MAV50. The difference between the performances of MAV20 and MAV50, as shown, is slight, which also indicates the ability of MAV algorithm to improve the performance by taking smaller step sizes in the interval (0, 5).

Tables 5.2 shows the number of better solutions provided by the algorithms comparing to MAV. Based on the results MAV provides 87.8, 93.9, and 73.5 percent of times better or equal than CDS for the Type 2 problems for generality levels 0.3, 0.6, and 0.9, respectively. Also, comparing to Gupta and Palmer, MAV provides better solution for more than 98 percent of times.
Figure 5.4

The percentage of the makespan values of the algorithms versus generality values comparing to MAV50 for medium (Type 2) problems.

Notice: Generality indicates the probability that a job skip a machine.
Table 5.2

The number of better makespan values of the algorithms comparing to MAV50 for medium (Type 2) problems with different generality values

<table>
<thead>
<tr>
<th>MAV50</th>
<th>Generality=0</th>
<th>Generality=0.3</th>
<th>Generality=0.6</th>
<th>Generality=0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>more</td>
<td>less</td>
<td>tie</td>
<td>more</td>
</tr>
<tr>
<td>Dan. RA</td>
<td>46</td>
<td>3</td>
<td>0</td>
<td>44</td>
</tr>
<tr>
<td>Gupta</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td>49</td>
</tr>
<tr>
<td>CDS</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>Palmer</td>
<td>45</td>
<td>0</td>
<td>3</td>
<td>47</td>
</tr>
</tbody>
</table>

The total number of the problems for each generality level = 49

The sizes of the problems \( n = 21 \) to \( 50 \) step 5; \( m = 21 \) to \( 50 \) step 5.

- **more**: the number of the schedules provided worse than MAV.
- **less**: the number of the schedules provided better than MAV.
- **tie**: the number of the schedules provided similar to MAV.

Notice: Generality indicates the probability that a job skip a machine.
In Figure 5.5 the relative CPU times for the algorithms to handle medium (Type 2) problems are shown. As shown the computational effort almost does not depend on the generality levels. The CPU time of MAV50 is more than the one of CDS, the average difference in CPU time between MAV and CDS is 16.3 seconds on the same cited computer. To compare the performance of MAV and CDS for almost the computational effort, we consider MAV 20. As shown in Figure 5.4, MAV20 is superior to CDS and requires less than 0.65 percent of the computational effort of CDS.

Large (Type 3) problems

In this part we consider the large problems (Type 3), which include problems having a large number of jobs to be processed in the large shops. The results for the makespan values of the algorithms, in percentage, comparing to the makespan value of the MAV algorithm have been shown in Figure 5.6. As shown the MAV algorithm provides better makespan values comparing to other algorithms. As the generality of the problems increases the relative performance of MAV gets much better than of other algorithms. In fact using MAV provides a saving of 4.06, 5.64, and 7.83 percent of shop time comparing to CDS for generality levels 0.3, 0.6 and 0.9, respectively.

The frequency of better solutions provided by the algorithms comparing to MAV50, is shown in Tables 5.3. Based on the results MAV provides 97.5, 93.8, and 86.4 percent of times better or equal solutions than CDS for generality levels 0.3, 0.6, and 0.9, respectively. Although the percentages of the number of better
solutions decreases from 97.5 to 86.4 for the respective generality levels 0.3 and 0.9, but the percentages of the corresponding the makespan percentages values increase from 4.06 to 7.83, which indicates that MAV50 provides schedules with much less makespan values as the generality levels increases.

Notice that in the computational results we have provided the performance of MAV10 and MAV20 for "small" and "medium" problems, respectively. The main reason for the introduction of MAV10 and MAV20 was to illustrate the better performance of MAV in a competitive CPU time versus CDS. Since for the large problems MAV50 provides solutions with both much smaller makespan and smaller CPU time, we have not include the performances of MAV with different step sizes. It is worthwhile to note that the makespan values provided by MAV20 is almost one percent more than MAV50.
Figure 5.5

The percentage of the running time values of the algorithms versus generality values comparing to MAV50 for medium (Type 2) problems.

Notice: Generality indicates the probability that a job skip a machine.
The percentage of the makespan values of the algorithms versus generality values comparing to MAV50 for large (Type 3) problems.

Notice: Generality indicates the probability that a job skip a machine.
Table 5.3

The number of better makespan values of the algorithms comparing to MAV50 for large (Type 3) problems with different generality values

<table>
<thead>
<tr>
<th>MAV50</th>
<th>Generality=0</th>
<th>Generality=0.3</th>
<th>Generality=0.6</th>
<th>Generality=0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>more less tie</td>
<td>more less tie</td>
<td>more less tie</td>
<td>more less tie</td>
</tr>
<tr>
<td>Dan. RA</td>
<td>76 5 0</td>
<td>79 2 0</td>
<td>76 5 0</td>
<td>70 11 0</td>
</tr>
<tr>
<td>Gupta</td>
<td>81 0 0</td>
<td>81 0 0</td>
<td>81 0 0</td>
<td>81 0 0</td>
</tr>
<tr>
<td>CDS</td>
<td>77 4 0</td>
<td>79 2 0</td>
<td>77 4 0</td>
<td>70 11 0</td>
</tr>
<tr>
<td>Palmer</td>
<td>77 3 1</td>
<td>79 2 0</td>
<td>79 1 1</td>
<td>79 0 2</td>
</tr>
</tbody>
</table>

The total number of the problems for each generality level= 81

The sizes of the problems $n = 51$ to $91$ step 5; $m = 51$ to $91$ step 5.

- **more**: the number of the schedules provided worse than MAV.
- **less**: the number of the schedules provided better than MAV.
- **tie**: the number of the schedules provided similar to MAV.

Notice: Generality indicates the probability that a job skip a machine.
In Figure 5.7 the CPU times for the algorithms to handle Type 3 problems are shown. As shown, for MAV and CDS the computational effort almost does not depend on the generality levels. The CPU time of MAV50 is 83.3 percent of the one of CDS, and by decreasing the number of the cited step sizes of \( \alpha \) in the interval \((0, 5)\) the CPU time can be much less than the existing value in the cost of a slight increase in the makespan value. The average difference of the CPU time in favor of MAV is 30.7 seconds and the maximum difference is 98.9 seconds on the same computer cited. Clearly, as the size of the shop and the number of the jobs to be scheduled increase the computational effort of MAV comparing to CDS decreases, meanwhile the number of better solutions and the percentage of the makespan value improves remarkably.
Figure 5.7
The percentage of the running time values of the algorithms versus generality values comparing to MAV50 for large (Type 3) problems
5.3.3 Flowshop with unbalanced load

As discussed before one of the problems of the functional algorithms is poor performance in the case of existing unbalanced loads. An unbalanced load is a processing time larger, e.g. ten times larger, than the average of all other processing times. In terms of statistical linear regression an unbalanced load of a job \( i \) is an outlier point in the \( t_{ij} \)-profile, defined in Chapter 2. In this part we examine the performance of the cited algorithms to handle pure flowshop problems having unbalanced loads for 10 percent of times. The average magnitude of the unbalanced load is considered to be ten times larger than the average processing times.

Three types of the problems are generated and solved by the algorithms. Since no significant difference in the computational efforts occurs, in Figure 5.8 and Table 5.4 the results of the makespan value percentages and the frequency of better solutions provided comparing to the MAV algorithm are presented, respectively.

As shown in Figure 5.8, MAV provides 4.09, 5.82, and 6.65 percents of saving in the shop time comparing to CDS for respective Types 1, 2, and 3; and, similarly, 6.09, 6.11, and 5.67 percents comparing to Palmer. The respective percentages for Gupta's method are 21.00, 24.52, and 21.68.

As we see the performance of the MAV algorithm for handling the problems with unbalanced load is very superior to all other algorithms. As the size of the shop and the number of the jobs increase the relative MAV performance also improves.
Figure 5.8

The percentage of the makespan values of the algorithms versus the size of the problems comparing to MAV50 for problems with unbalanced loads.
In Table 5.4 the percentage of better solutions provided by the cited algorithms comparing to MAV are presented. As shown, the percentage of better or equal solution remarkably improves as the shop size and the number of the jobs to be scheduled increase. In the overall result, CDS has shown to be superior to all other algorithms, and MAV is better than CDS more than 80 percent of times.

5.3.4 Comparing MAV to NEH

As cited in the literature review NEH algorithm (Newaz et al., 1983) is one of the semi-exhaustive heuristic procedures, which has been compared to the cited algorithms for the small (Type 1) problems. The main inherent difficulty of NEH procedure is the tremendous computational efforts required for the problems of Type 2 and 3. For instance, the average CPU time on a Personal Computer AT 386 with co-processor for NEH to handle a flowshop with m=90 and n=90 is 6511 seconds (1:48:31 hour) where the average CPU time for MAV50 is 232 seconds (0:03:52 hour). Thus, the computational effort of NEH is 28 times greater than that of MAV. For small (Type 1) problems, NEH algorithm provided 0.81 percent makespan better than MAV; and also provided 59.77 of times better solutions than MAV.

Based on the results, although MAV procedure is a functional one, it can compete with NEH in dealing with makespan flowshop problems. In fact using MAV instead of NEH provides 0.81 percent increase in the makespan, which is very slight. Because of the tremendous computational effort, NEH is not efficient
to be applied to large problems and also for dynamic flowshop problems, MAV can be considered as a strong tool for the general flowshop scheduling.

In the next subsection the computational complexity of MAV, CDS, and NEH will be analyzed and discussed.

5.3.5 Computational complexity

In this study we consider the rate of the growth in solution time requirements as the size of the problem increases. The computational requirement is estimated by considering the frequency of each elementary computational step in an algorithm. It is supposed that the time for the elementary computational steps is fixed. Based on the study, a time complexity function (TCF) for each algorithm can be obtained, which indicates the upper bound on the total frequency over all computational steps. In fact TCF indicates the maximum computation time required to solve a given problem. In other terms, TCF is a worse-case criterion of the an algorithm, for a detailed discussion see Garey and Johnson, (1979).

For a general flowshop with m machines and n jobs the TCF for MAV is \( \Theta(T) = (s)(n)(n+m) \), where s is the number of step sizes in the interval (0,5). For instance, for MAV50 s is equal to 50. The TCF is a polynomial one, and the maximum computational time directly depends on the number of the step sizes s.

The TCF's for the CDS algorithm is \( \Theta(T) = (m)(n)(n+m) \); and for NEH is \( \Theta(T) = (1/2)(m)(n^3) \). As we see the TCF's for other two algorithms are also polynomial.
Table 5.4

The number of better makespan values of the algorithms comparing to MAV50 for various sizes of problems with unbalanced load

<table>
<thead>
<tr>
<th>MAV50</th>
<th>Type 1 (%)</th>
<th>Type 2 (%)</th>
<th>Type 3 (%)</th>
<th>Overall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>more</td>
<td>less</td>
<td>tie</td>
<td>more</td>
</tr>
<tr>
<td>Dan. RA</td>
<td>68.75</td>
<td>30.47</td>
<td>.08</td>
<td>83.67</td>
</tr>
<tr>
<td>Gupta</td>
<td>86.82</td>
<td>11.33</td>
<td>1.95</td>
<td>100</td>
</tr>
<tr>
<td>CDS</td>
<td>64.06</td>
<td>33.98</td>
<td>1.95</td>
<td>87.76</td>
</tr>
<tr>
<td>Palmer</td>
<td>75.00</td>
<td>1.95</td>
<td>24.22</td>
<td>89.90</td>
</tr>
</tbody>
</table>

The total number of the problems for Types 1, 2, and 3 are 256, 49, and 81, and the sizes of the problems of each type are given in Tables 5.1, 5.2, and 5.3.

more: the percentage of the schedules provided worse than MAV.

less: the percentage of the schedules provided better than MAV.

tie: the percentage of the schedules provided similar to MAV.
The TCF's of the MAV and CDS algorithms indicate that for a fixed value of \( s \leq m \) the worse-case of MAV is preferred to or is similar to CDS. To compare MAV to NEH, let \( n=m \), for MAV it holds \( \Theta(T)=(2)(s)(n^2) \) and for NEH, it holds \( \Theta(T)=(1/2)n^4 \). As long as \( s \leq (1/4)(n^2) \), the worse-case of MAV is preferred to or is similar to NEH. For instance, for \( s=20 \), for \( m=n \geq 9 \), MAV is superior from the complexity aspect to NEH.

We can conclude that for a fixed \( s \), say \( s=50 \), and for large problems MAV is computationally preferred to CDS and NEH.

5.4 Summary and conclusions

5.4.1 Summary

In this chapter the MAV algorithm for scheduling general flowshop problem is introduced, which contained an unknown parameter \( \alpha \). The frequency distribution of the parameter is determined experimentally by solving more than 700 problems of different sizes with random generated processing times.

Then, for a precise analysis of the performance of the algorithms, the problems are grouped in several types, depending upon the size of shops and the number of the jobs to be scheduled. The performances of MAV has been compared to CDS, Palmer, Gupta, RA of Dannenberg for all sizes; and also compared to NEH for the problem having up to 20 machines and/or jobs. The algorithm are applied to general flowshop problems having different generality levels and also unbalanced load. The
comparison criteria are the makespan value, the number of better solutions, and the required computational efforts of each algorithm.

The comparisons are done for various levels of generalities, the probability of a job does not need a certain machine. Also, the performances of the algorithms has be examined for the case unbalanced loads exist.

Finally, the computational complexity of MAV, CDS, and NEH has been determined and comparatively analyzed.

5.4.2 Conclusions

The experimental results from the comparison of MAV to CDS, Palmer, Gupta, RA of Dannenberg for all types of the problem have shown that MAV provides remarkably better makespan and also the much higher frequency of the better solutions. The experiments have shown that the best algorithm among CDS, Palmer, Gupta, RA of Dannenberg is CDS and the worse is Gupta's algorithm. On the average MAV provides 4.91 and 20.38 percent better makespan than CDS and Gupta, respectively. Also, in average MAV provides 79.67 and 97.86 of times better than or similar to CDS and Gupta, respectively. As the size, generality, and the load unbalancicity increase the performance of MAV is superior to the cited algorithms.

MAV is computationally superior to both CDS and NEH for most sizes of the problems, and the superiority increases as the problems get
larger. For instance, for large problems, e.g. n=m=90, the average CPU time of the NEH algorithm exceeds 1:48:31 hours, where of the MAV is 0:03:52. In other term NEH requires a CPU time 28.04 times more than the one of MAV. Meanwhile, for the problems having up to 20 jobs and/or machine the makespan provided by MAV is 0.86 percent more than NEH, which is slight.
CHAPTER 6

CONCLUSIONS AND SUGGESTION FOR FUTURE RESEARCH

6.1 Conclusions

In addition to the conclusions provided at the end of each chapter, in this research we have concluded that multiattribute value theory is a strong tool for providing a new functional approach to the modeling of the traditional general makespan flowshop problem. The performance of the resulted algorithm, which are the makespan, the number of better solutions, and the required computational effort, for different conditions and various problems have been superior to the existing applied functional and non-functional algorithms, and even compete with the semi-exhaustive procedures.

The great advantages of the new approach is to provide a new theoretical background for the problem, which is apt to decrease a set of the simplifying assumptions. This advantage helps to decrease the existing gap between theory and application, which has been a challenging matter in sequential scheduling. Another advantage of the new approach is providing the first functional approach, with all privilege of the functional methods such as ease of computation, applying to jobs separately, and so on; meanwhile, its performances is superior to nonfunctional ones.
6.2 Suggestions for future research

This work can be extended in at least four areas. The first is to examine the parameter $\alpha$, which has been evaluated experimentally. Examples of possible investigations include possible relationships between $\alpha$ and/or $m$ and $n$, the differences of the problems having large $\alpha$'s (say $\alpha > 50$) from the ones having small $\alpha$'s (say $\alpha < 5$), and the sensitivity of the objective value versus $\alpha$.

The next area is modifying the approach for non-traditional, real flowshop problems such as flexible one, which may have parallel processors of each type, and also for dynamic flowshops, in which a new job may come to the shop and/or be cancelled from further processing. An extension of this approach applied for repetitive flowshop is given in Ramazani and Younis (1991).

The third area is extending the method for the job shop problems. The essence of this extension can be presenting job shops in terms of a proper general flowshop and modifying the method to solve the new problem.

The last area is the introduction of uncertainty and imprecision incidents such as breakdowns, and or fuzzy network techniques (Kaufmann and Gupta, 1988) to reduce the uncertain case to the case investigated herein. Also, in case of imperfect information of the processing times, using fuzzy regression technique applied in Ramazani and Duckstein (1991) provides a new formulation for the problem.
REFERENCES


- Dudek, R.A. and Teuton, O.F. "Development of M-Stage Decision Rule for Scheduling n Jobs through m-machines" Ops. Res. vol. 12, pp. 397-411 (1964)


- Glover, F., "Tabu (Taboo) Search: Part II" ORSA J. Comp. vol. 2, no. 1, (1990)


- Gorman, W. M. "Conditions for Additive Separability" Econometrica vol. 36, pp. 605-609 (1968(ii))


- Keuls, M. "The Use of the Studentized Range in Connection with the Analysis of Variance" Euphytica, vol. 1, pp. 112-122 (1952)

- Klockars, J.R and Sax, Multiple Comparisons, Beverly Hills, Sage Publication (1986)


- Ramazani, R. and L. Duckstein "Fuzzy analysis of the effect of university research on regional technologies" in Kacprzyk, J. and M.


- Wiede, W. Jr., An Interactive Scheduling System for the Operation of Multi-Product Plants, PhD Dissertation, Purdue University, Laffayet, Indiana (1984)


APPENDIX 1

**Theorem 3.2** Given an $m \times n$ general flowshop problem with $[t_{i,j}], \ i=1,...,n; j=1,...,m$, and a sequence $(1,2,...,i,i+1,...,n)$, which indicates the sequence of the jobs, e.g., $J_{i}$ precedes $J_{i+1}$. Let the makespan for the given sequence be denoted by $T$, and for the reverse problem with the reverse sequence be denoted by $T^{R}$. Then, $T = T^{R}$.

**Proof:** By induction, for $k = 2$, it has proved in Theorem 3.1. Let the theorem be true for $k = n$. That is, we have two sequences, which are $S = (1,2,...,n)$ and $S^{R} = (n,...,2,1)$ such that

$$T_{S} = T_{S}^{R}. \quad (A1.1)$$

We construct a new job, say $J_{e}$, such that to be equivalent to the set of the $n$ jobs with the given sequence. This means $J_{e}$ preserves both the processing times and the given sequence order. The processing times of $J_{e}$ are as follow:

$$t_{e,1} = F_{n,1}$$

and

$$t_{e,j} = F_{n,j} - F_{n,j-1} \quad j = 2,3,...,m.$$ 

The makespan of $J_{e}$ if it be processed alone will be
\[ T_e = \sum_{j=1}^{m} (t_{e,j}) = F_{n,m} \quad \text{(A1.2)} \]

By the definition of the flowtime

\[ T_s = F_{n,m} \quad \text{(A1.3)} \]

Also, if we reverse \( J_e \), the corresponding processing times of the reverse of \( J_e \) will be

\[ t_{e,m} = F_{n,1} \]

and

\[ t_{e,m-j+1} = F_{n,j} - F_{n,j-1}, j = 2,3,\ldots,m. \]

Then,

\[ T^R_e = \sum_{j=1}^{m} (t_{e,j}) = F_{n,m} \quad \text{(A1.4)} \]

Since, by (A1.1), it is assumed \( T_s = T^R_s \)

\[ T^R_e = T^R_s \quad \text{(A1.5)} \]

Finally, by (A1.5) it concludes that \( J_e \) and \( J^R_e \) represent the original and the reverse problems, respectively.
Now, we continue the induction process by introducing the \((n+1)\)th job, i.e., \(J_{n+1}\). The augmented original problem will have \((n+1)\) jobs and so is the augmented given sequence \(S = (1,2,...,n+1)\). Similarly, it holds for the reverse problem with the reverse augmented sequence \(S^R = (n+1,...,2,1)\). We use the equivalence \(J_e\) and \(J^R_e\), as concluded in (A1.2) and (A1.4), which result in using \(s = (e, n+1)\) and \(s^R = (n+1, e^R)\) instead of \(S\) and \(S^R\), respectively. Therefore, the original and the reverse problems will be reduced to an \(m \times 2\) problem, then, by Theorem 3.1, it can be concluded that,

\[
T_S = T^R_S.
\]

Q.E.D.

**An Illustrative example:**

The processing times matrix for the original \(5 \times 4\) flowshop problem is as follows

<table>
<thead>
<tr>
<th></th>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(M_3)</th>
<th>(M_4)</th>
<th>(M_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(J_1)</td>
<td>17</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>21</td>
</tr>
<tr>
<td>(J_2)</td>
<td>9</td>
<td>13</td>
<td>10</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td>(J_3)</td>
<td>9</td>
<td>4</td>
<td>55</td>
<td>8</td>
<td>61</td>
</tr>
<tr>
<td>(J_4)</td>
<td>4</td>
<td>8</td>
<td>18</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

The makespan values for the original sequences \((J_4, J_3, J_2, J_1)\) and \((J_4, J_1, J_3, J_2)\) are 193 and 180, respectively. Then, processing times matrix for the corresponding reverse is as follows:
<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>$M_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_{R_1}$</td>
<td>21</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>17</td>
</tr>
<tr>
<td>$J_{R_2}$</td>
<td>18</td>
<td>1</td>
<td>10</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>$J_{R_3}$</td>
<td>61</td>
<td>8</td>
<td>55</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>$J_{R_4}$</td>
<td>2</td>
<td>7</td>
<td>18</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

The makespan values for the reverse problem $(J_{R_1}, J_{R_2}, J_{R_3}, J_{R_4})$ and $(J_{R_2}, J_{R_3}, J_{R_1}, J_{R_4})$, which are the corresponding reverse problems of the cited original sequences, are 193 and 180, respectively.
APPENDIX 2

As cited in subsection (4.3.2.1), this appendix considers the validity of the additivity of the MAV. For this purpose we extensively use the formulas for the idle times of the machines, which are introduced in (4.7), and of the makespan for a sequence, provided in (4.5) and (4.6). For convenience, we defer to recall them frequently. Notice that as explained in the subsection, without loss of generality, it is assumed that \((h,i)\) is the optimal sequence. For each case a feasibility study is done. The cases are feasible unless it is mentioned as infeasible. Then, we consider the validity of the additivity assumption. When \(T_h < T_i\) holds the assumption is valid; otherwise, the assumption is violated. We remind that \(t_{i,q} = t_{h,q}\), for \(q = e, p, k+1\). The constraints for part \((A)_{i}\) are given in (A-1) thru (A-5).

\[
\begin{align*}
& t_{i,j} + t_{i,k} > t_{h,j} + t_{h,k} \quad \text{(A-1)} \\
& t_{i,j} > t_{h,k} \quad \text{(A-2)} \\
& t_{i,k} < t_{h,k} \quad \text{(A-3)} \\
& t_{i,k} > t_{h,j} \quad \text{(A-4)}
\end{align*}
\]

By (A-1) thru (A-4) \(=> t_{h,j} < t_{i,j}\) \(\text{(A-5)}\)

\[
\begin{align*}
\text{I - } \\
& I^h_{i,j} = 0 \Rightarrow t_{i,e} - t_{h,j} \leq 0 \Rightarrow t_{i,e} \leq t_{h,j} \quad \text{(A-6)} \\
& I^i_{h,i} = 0 \Rightarrow t_{h,e} - t_{i,j} \leq 0 \Rightarrow t_{h,e} \leq t_{i,j} \quad \text{(A-7)}
\end{align*}
\]

\[
\begin{align*}
\text{II - } \\
& I^h_{i,j} > 0 \Rightarrow t_{i,e} - t_{h,j} > 0 \Rightarrow t_{i,e} > t_{h,j} \quad \text{(A-8)}
\end{align*}
\]
I^h_{i,j} = 0 \implies t_{i,e} - t_{i,j} \leq 0 \implies t_{i,e} \leq t_{i,j} \quad (A-7)

III -
I^h_{i,j} = 0 \implies t_{i,e} - t_{h,j} \leq 0 \implies t_{i,e} \leq t_{h,j} \quad (A-6)
I^i_{h,i} > 0 \implies t_{h,e} - t_{i,j} > 0 \implies t_{h,e} > t_{i,j} \quad (A-9)

By (A-5) and (A-8) \implies t_{i,j} < t_{h,j} \text{, which contradicts (A-5) } \implies \text{ infeasible}

IV -
I^h_{i,j} > 0 \implies t_{i,e} - t_{h,j} > 0 \implies t_{i,e} > t_{h,j} \quad (A-8)
I^i_{h,i} > 0 \implies t_{h,e} - t_{i,j} > 0 \implies t_{h,e} > t_{i,j} \quad (A-9)

All combinations for (I) are as follows:

I- i
I^h_{i,p} > 0 \implies t_{i,j} - t_{h,p} > 0 \implies t_{i,j} > t_{h,p} \quad (A-10)
I^i_{h,p} > 0 \implies t_{h,j} - t_{i,p} > 0 \implies t_{h,j} > t_{i,p} \quad (A-11)

I- i - a
I^h_{i,k} > 0 \implies t_{i,j} - t_{h,k} > 0 \implies t_{i,j} > t_{h,k} \quad (A-12)
I^i_{h,k} > 0 \implies t_{h,j} - t_{i,k} > 0 \implies t_{h,j} > t_{i,k} \quad (A-13)

(A-13) contradicts (A-4) \implies \text{ infeasible}

I- i - b
I^h_{i,k} = 0 \implies t_{i,j} - t_{h,k} \leq 0 \implies t_{i,j} < t_{h,k} \quad (A-14)
I^i_{h,k} = 0 \implies t_{h,j} - t_{i,k} \leq 0

(A-14) contradicts (A-2) \implies \text{ infeasible}

I- i - c
I^h_{i,k} > 0 \implies t_{i,j} - t_{h,k} > 0 \quad (A-12)
I^i_{h,k} = 0 \implies t_{h,j} \leq t_{i,k} \quad (A-15)

(A-15) and (A-4) result in \quad t_{h,j} < t_{i,k}

I- i - c - 1
I^h_{i,k+1} = 0 \implies t_{i,j} - t_{h,k+1} \leq 0 \implies t_{i,j} \leq t_{h,k+1} \quad (A-16)
\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \Rightarrow t_{h,k} \leq t_{i,k+1} \quad (A-17) \]
\[ T_i - T_h = t_{i,j} + t_{i,k} + t_{h,k+1} - t_{h,k} + t_{h,k+1} + t_{i,k+1} \]
By (A-16) and (A-17), \( T_i - T_h > 0 \)
\[ \Rightarrow T_h < T_i \]

**I- i - c - 2**
\[ I^h_{i,k+1} > 0 \Rightarrow t_{i,j} - t_{h,k+1} > 0 \]
\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \]
\[ \Rightarrow T_h < T_i \]

**I- i - c - 3**
\[ I^h_{i,k+1} = 0 \Rightarrow t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,k} > t_{i,k+1} \]
\[ \Rightarrow T_h < T_i \]

**I- i - c - 4**
\[ I^h_{i,k+1} > 0 \Rightarrow t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,k} > t_{i,k+1} \]
\[ \Rightarrow T_i - T_h = t_{h,k} + t_{h,k+1} - t_{h,j} - t_{i,k+1} \]
Since \( t_{h,k} > t_{h,j} \Rightarrow T_h < T_i \)

**I- i - d**
\[ I^h_{i,k} = 0 \Rightarrow t_{i,j} - t_{h,k} \leq 0 \quad (A-18) \]
\[ I^i_{h,k} > 0 \Rightarrow t_{h,j} - t_{i,k} > 0 \Rightarrow t_{h,j} < t_{i,k} \]
\( (A-18) \) contradicts (A-2) \( \Rightarrow \) infeasible

**I- ii**
\[ I^h_{i,p} > 0 \Rightarrow t_{i,j} - t_{h,p} > 0 \Rightarrow t_{i,j} > t_{h,p} \quad (A-19) \]
\[ I^i_{h,p} = 0 \Rightarrow t_{h,j} - t_{i,p} \leq 0 \Rightarrow t_{h,j} \leq t_{i,p} \quad (A-20) \]

**I- ii - a**
\[ I^h_{i,k} > 0 \Rightarrow t_{i,j} - t_{h,k} > 0 \]
\[ I^i_{h,k} > 0 \implies t_{h,p} \cdot t_{i,k} > 0 \]

**I- ii - a - 1**

\[
I^h_{i,k+1} = 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \quad (A-21)
\]

\[
I^i_{h,k+1} = 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} \leq 0 \quad (A-22)
\]

By (A-1), (A-21) and (A-22) \( T_h < T_i \)

**I- ii - a - 2**

\[
I^h_{i,k+1} > 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \quad (A-23)
\]

\[
I^i_{h,k+1} = 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} \leq 0 \quad (A-24)
\]

\[ T_i \cdot T_h = t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{i,j} - t_{i,k} + t_{h,k} + t_{h,k+1} \]

By (A-3), (A-4), (A-23), and (A-24) \( T_i - T_h > 0 \)

\[ T_h < T_i \]

**I- ii - a - 3**

\[
I^h_{i,k+1} = 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0
\]

\[
I^i_{h,k+1} > 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0
\]

\[ T_i - T_h = t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} + t_{h,p} + t_{h,k} - t_{i,j} - t_{i,k} > 0 \]

\[ T_h < T_i \]

**I- ii - a - 4**

\[
I^h_{i,k+1} > 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0
\]

\[
I^i_{h,k+1} > 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0
\]

\[ T_i - T_h = t_{h,p} + t_{h,k} - t_{h,j} - t_{i,k} > 0 \]

\[ T_h < T_i \]

**I- ii - b**

\[
I^h_{i,k} = 0 \implies t_{i,j} + t_{i,p} - t_{h,p} - t_{h,k} \leq 0 \quad (A-25)
\]

\[
I^i_{h,k} = 0 \implies t_{h,j} - t_{i,p} \leq 0
\]

(A-25) contradicts (A-2) \( \implies \) infeasible
I- ii - c

\[ I^h_{i,k} > 0 \implies t_{i,j} - t_{h,k} > 0 \]
\[ I^l_{h,k} = 0 \implies t_{h,p} - t_{i,k} \leq 0 \]  
(A-26)

I- ii - c - 1

\[ I^h_{i,k+1} = 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{i,k+1} \leq 0 \]
\[ I^l_{h,k+1} = 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} \leq 0 \]

\[ T_i - T_h = t_{i,j} + t_{i,k} + 0 - t_{h,j} - t_{i,k} - 0 > 0 \]
\[ T_h < T_i \]

I- ii - c - 2

\[ I^h_{i,k+1} > 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I^l_{h,k+1} = 0 \implies t_{h,k} - t_{i,k+1} \leq 0 \]

\[ T_i - T_h = t_{i,j} + t_{i,k} + 0 - t_{h,j} - t_{i,k} - t_{i,k} + t_{h,k} + t_{h,k+1} > 0 \]
\[ T_h < T_i \]

I- ii - c - 3

\[ I^h_{i,k+1} = 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I^l_{h,k+1} > 0 \implies t_{h,k} - t_{i,k+1} \leq 0 \]

\[ T_i - T_h = t_{i,j} + t_{i,k} + I^l_{h,k+1} - t_{h,j} - t_{i,k} + 0 > 0 \]
\[ T_h < T_i \]

I- ii - c - 4

\[ I^h_{i,k+1} > 0 \implies t_{i,j} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I^l_{h,k+1} > 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0 \]

\[ T_i - T_h = t_{h,k} - t_{h,j} > 0 \]
\[ T_h < T_i \]

I- ii - d

\[ I^h_{i,k} = 0 \implies t_{i,j} - t_{h,k} \leq 0 \]  
(A-27)
\[ I^l_{h,k} > 0 \implies t_{h,p} - t_{i,k} > 0 \]

(A-27) contradicts (A-2) => infeasible
\begin{align*}
I_{i,p}^h = 0 & \Rightarrow t_{i,j} - t_{h,p} \leq 0 \Rightarrow t_{i,j} \leq t_{h,p} \quad (A-28) \\
I_{h,p}^i > 0 & \Rightarrow t_{h,j} - t_{i,p} > 0 \Rightarrow t_{h,j} > t_{h,p} \quad (A-29)
\end{align*}

since $t_{h,p} = t_{i,p}$, then, (A-28) and (A-29) contradict (A-5)

$\Rightarrow$ infeasible

\begin{align*}
I_{i,p}^h = 0 & \Rightarrow t_{i,j} - t_{h,p} \leq 0 \Rightarrow t_{i,j} \leq t_{h,p} \quad (A-30) \\
I_{h,p}^i = 0 & \Rightarrow t_{h,j} - t_{h,p} \leq 0 \Rightarrow t_{h,j} \leq t_{i,p}
\end{align*}

\begin{align*}
I_{i,k}^h > 0 & \Rightarrow t_{i,j} - t_{h,k} > 0 \Rightarrow t_{i,j} > t_{h,k} \quad (A-31) \\
I_{h,k}^i > 0 & \Rightarrow t_{h,p} - t_{i,k} > 0 \Rightarrow t_{h,p} > t_{i,k} \quad (A-32)
\end{align*}

\begin{align*}
I_{i,k+1}^h = 0 & \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \\
I_{h,k+1}^i = 0 & \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} \leq 0
\end{align*}

$\Rightarrow T_h < T_i$

\begin{align*}
I_{i,k+1}^h > 0 & \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \quad (A-33) \\
I_{h,k+1}^i = 0 & \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} \leq 0 \quad (A-34)
\end{align*}

(A-33) and (A-34) result in $t_{i,k} \leq t_{h,k}$, which contradicts (A-3)

$\Rightarrow$ infeasible

\begin{align*}
I_{i,k+1}^h = 0 & \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \\
I_{h,k+1}^i > 0 & \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0
\end{align*}

$\Rightarrow T_h < T_i$

\begin{align*}
I_{i,k+1}^h > 0 & \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \\
I_{h,k+1}^i > 0 & \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0
\end{align*}
\[ T_i - T_h = t_{i,j} + t_{h,k} - t_{i,k} - t_{h,j} > 0 \]
\[ T_h < T_i \]

I- iv - c

\[ I^{h}_{i,k} > 0 \implies t_{i,j} - t_{h,k} > 0 \implies t_{i,j} > t_{h,k} \quad (A-35) \]
\[ I^{i}_{h,k} = 0 \implies t_{h,p} - t_{i,k} \leq 0 \implies t_{h,p} \leq t_{i,k} \]

(A-35) contradicts (A-30) \implies \text{infeasible}

I- iv - d

\[ I^{h}_{i,k} > 0 \implies t_{i,j} - t_{h,k} > 0 \implies t_{i,j} > t_{h,k} \quad (A-36) \]
\[ I^{i}_{h,k} > 0 \implies t_{h,p} - t_{i,k} > 0 \implies t_{h,p} > t_{i,k} \quad (A-37) \]

By (A-30)(A-36)(A-37), \( t_{i,j} < t_{h,k} \), which contradicts (A-2)

\[ \implies \text{infeasible} \]

All combinations for (II) are as follows:

II- i

\[ I^{h}_{i,p} > 0 \implies t_{i,e} + t_{i,j} - t_{h,j} - t_{h,p} > 0 \quad (A-38) \]
\[ I^{i}_{h,p} > 0 \implies t_{h,j} - t_{i,p} > 0 \implies t_{h,j} > t_{i,p} \quad (A-39) \]

II- i - a

\[ I^{h}_{i,k} > 0 \implies t_{i,e} + t_{i,j} - t_{h,j} + t_{h,p} > 0 \quad (A-40) \]
\[ I^{i}_{h,k} > 0 \implies t_{h,j} + t_{h,p} - t_{i,p} + t_{i,k} > 0 \quad (A-41) \]

since \( t_{h,p} = t_{i,p} \), (A-41) indicates that since \( t_{h,j} < t_{i,k} \),

which contradicts (A-4) \implies \text{infeasible}

II- i - b

\[ I^{h}_{i,k} = 0 \implies t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} < 0 \implies t_{i,e} < t_{h,k} \quad (A-42) \]
\[ I^{i}_{h,k} = 0 \implies t_{h,j} + t_{h,p} - t_{i,p} + t_{i,k} < 0 \implies t_{h,j} \leq t_{i,k} \quad (A-43) \]

II- i - b - 1

\[ I^{h}_{i,k+1} = 0 \implies t_{i,k} - t_{h,k+1} \leq 0 \implies t_{i,k} \leq t_{h,k+1} \]
\[ I^{i}_{h,k+1} = 0 \implies t_{h,k} - t_{i,k+1} \leq 0 \]
\[ T_h < T_i \]

**II- i - b - 2**

\[ I^h_{i, k+1} > 0 \Rightarrow t_{i, k} - t_{h, k+1} > 0 \Rightarrow t_{i, k} > t_{h, k+1} \] (A-44)

\[ I^i_{h, k+1} = 0 \Rightarrow t_{h, k} - t_{i, k+1} \leq 0 \Rightarrow t_{h, k} \leq t_{i, k+1} \] (A-45)

(A-44) and (A-45) contradicts (A-3) \( \Rightarrow \) infeasible

**II- i - b - 3**

\[ I^h_{i, k+1} = 0 \Rightarrow t_{i, k} - t_{h, k+1} \leq 0 \Rightarrow t_{i, k} \leq t_{h, k+1} \]

\[ I^i_{h, k+1} > 0 \Rightarrow t_{h, k} - t_{i, k+1} > 0 \Rightarrow t_{h, k} > t_{i, k+1} \]

\[ T_i - T_h = t_{i, j} + t_{i, k} + I^i_{h, k+1} - t_{h, j} - t_{h, k} > 0 \]

\[ \Rightarrow T_h < T_i \]

**II- i - b - 4**

\[ I^h_{i, k+1} > 0 \Rightarrow t_{i, k} - t_{h, k+1} > 0 \Rightarrow t_{i, k} > t_{h, k+1} \]

\[ I^i_{h, k+1} > 0 \Rightarrow t_{h, k} - t_{i, k+1} > 0 \Rightarrow t_{h, k} > t_{i, k+1} \]

\[ T_i - T_h = t_{i, j} + t_{i, k} + t_{i, k} - t_{h, k+1} - t_{h, j} - t_{h, k} - t_{h, k} - t_{i, k+1} \]

since \( t_{h, k} > t_{i, k} \) \( \Rightarrow \) \( T_h < T_i \)

**II- i - c**

\[ I^h_{i, k} > 0 \Rightarrow t_{i, e} + t_{i, j} - t_{h, j} + t_{h, k} > 0 \]

\[ I^i_{h, k} = 0 \Rightarrow t_{h, j} - t_{i, k} \leq 0 \Rightarrow t_{h, j} \leq t_{i, k} \] (A-46)

**II- i - c - 1**

\[ I^h_{i, k+1} = 0 \Rightarrow t_{i, e} + t_{i, j} - t_{h, j} + t_{h, k} \leq 0 \]

\[ I^i_{h, k+1} = 0 \Rightarrow t_{h, k} - t_{i, k+1} \leq 0 \Rightarrow t_{h, k} - t_{i, k+1} \leq 0 \]

\[ \Rightarrow \] \( T_h < T_i \)

**II- i - c - 2**

\[ I^h_{i, k+1} > 0 \Rightarrow t_{i, e} + t_{i, j} + t_{i, k} - t_{h, j} - t_{h, k} - t_{h, k+1} > 0 \]

\[ I^i_{h, k+1} = 0 \Rightarrow t_{h, k} - t_{i, k+1} \leq 0 \]

\[ \Rightarrow T_i - T_h = t_{h, k+1} - t_{i, e} \]
Notice that $t_{i,j} \geq t_{i,e} > t_{h,j}$, therefore,

if $t_{h,k+1} > t_{h,k}$, then, $T_h < T_i$.

II- i - c - 3

$I^h_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0$

$I^i_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \Rightarrow t_{h,k} > t_{i,k+1}$

$\Rightarrow T_h < T_i$

II- i - c - 4

$I^h_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0$

$I^i_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0$

$\Rightarrow T_i - T_h = t_{h,k} + t_{i,k+1} + t_{h,k+1} - t_{i,e} = t_{h,k} - t_{i,e}$

Therefore, if $t_{h,k} > t_{i,e}$, then, $T_h < T_i$

II- i - d

$I^h_{i,k} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} > 0$

$I^i_{h,k} = 0 \Rightarrow t_{h,j} - t_{i,k} \leq 0 \Rightarrow t_{h,j} \leq t_{i,k}$ \hspace{1cm} (A-47)

(A-46) contradicts (A-5) $\Rightarrow$ infeasible

II- ii

$I^h_{i,p} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} - t_{h,p} > 0$

$I^i_{h,p} = 0 \Rightarrow t_{h,e} + t_{h,j} - t_{i,j} - t_{i,p} \leq 0$

II- ii - a

$I^h_{i,k} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} > 0$

$I^i_{h,k} > 0 \Rightarrow t_{h,p} - t_{i,k} > 0 \Rightarrow t_{h,p} > t_{i,k}$ \hspace{1cm} (A-48)

II- ii - a - 1

$I^h_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} \leq 0$

$\Rightarrow t_{i,e} < t_{h,k}$

$I^i_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0$

$\Rightarrow T_h < T_i$
II- ii - a - 2  \( I^h \), \( k+1 > 0 \)  \( \Rightarrow t_{i,c} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} \leq 0 \)  

(A-49) 

\( I^l_{h,k+1} = 0 \)  \( \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \)  

(A-50) 

\[ T_i - T_h = t_{h,k+1} - t_{i,e} \]  

Notice that by (A-7), (A-48), (A-49) and (A-50),  
\[ t_{i,j} > t_{i,e} > t_{h,j} \quad \text{and} \quad t_{h,k+1} > t_{h,k} \]  

If  \( t_{h,k+1} > t_{i,e} \), then,  \( T_h < T_i \).  

II- ii - a - 3  \( I^h \), \( k+1 = 0 \)  \( \Rightarrow t_{i,c} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} \leq 0 \)  

\( I^l_{h,k+1} > 0 \)  \( \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} > 0 \)  

\[ T_h < T_i \]  

II- ii - a - 4  \( I^h \), \( k+1 > 0 \)  \( \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} \leq 0 \)  

\( I^l_{h,k+1} > 0 \)  \( \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} > 0 \)  

\[ T_i - T_h = t_{h,p} + t_{h,k} - t_{i,k} - t_{i,e} \]  

Notice that  \( t_{i,j} > t_{i,e} > t_{h,j} \)  and  \( t_{h,k} > t_{i,k} \), and by (A-48),  
\[ t_{h,k} > t_{i,k} \]  

If  \( t_{h,k+1} > t_{i,e} \), then,  \( T_h < T_i \).  

II- ii - b  \( I^h = 0 \)  \( \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} \leq 0 \)  \( \Rightarrow t_{i,e} \leq t_{h,j} \)  

(A-51) 

\( I^l_{h,k} = 0 \)  \( \Rightarrow t_{h,p} - t_{i,k} \leq 0 \)  \( \Rightarrow t_{h,p} \leq t_{i,k} \)  

(A-51) contradicts (A-8) \( \Rightarrow \) infeasible 

II- ii - c  \( I^h = 0 \)  \( \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} > 0 \)  

\( I^l_{h,k} = 0 \)  \( \Rightarrow t_{h,p} - t_{i,k} \leq 0 \)  \( \Rightarrow t_{h,p} \leq t_{i,k} \)  

(A-52)
II- ii - c - 1 \[ I_{i,k+1}^h = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ \Rightarrow t_{i,e} < t_{h,k} \]
\[ I_{i,k+1}^l = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]
\[ \Rightarrow T_h < T_i \]

II- ii - c - 2 \[ I_{i,k+1}^h > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I_{i,k+1}^l = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \]
\[ \Rightarrow T_i - T_h = t_{h,k+1} - t_{i,e} \]

II- ii - c - 3 \[ I_{i,k+1}^h = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ I_{i,k+1}^l > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \]
\[ \Rightarrow T_h < T_i \]

II- ii - c - 4 \[ I_{i,k+1}^h > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I_{i,k+1}^l > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \]
\[ \Rightarrow T_i - T_h = t_{h,k} - t_{i,e} \]

II- ii - d \[ I_{h,k}^i = 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} \leq 0 \]
\[ \Rightarrow t_{i,e} \leq t_{h,k} \] (A-53)
\[ I_{h,k}^l > 0 \Rightarrow t_{h,p} - t_{i,k} > 0 \Rightarrow t_{h,p} > t_{i,k} \] (A-54)

By (A-8), (A-53) and (A-54), \( t_{h,k} > t_{i,j} \), which

contradicts (A-3) \Rightarrow \text{infeasible}

II- iii \[ I_{i,p}^h = 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} \leq 0 \] (A-55)
\[ I^i_{h, p} > 0 \implies t_{h, i} - r_{i, p} > 0 \implies t_{h, j} > t_{i, p} \quad (A-56) \]

By (A-4), (A-55) and (A-56), \( t_{h, k} > t_{i, k} \), which contradicts (A-2) => infeasible

\[
\begin{align*}
\text{II- iv} & \quad I^h_{i, p} = 0 \implies t_{i, e} + t_{i, j} - t_{h, i} - t_{h, p} \leq 0 \\
& \implies t_{i, e} \leq t_{h, p} \quad (A-57) \\
I^h_{i, p} = 0 & \implies t_{h, j} - t_{i, p} \leq 0 \implies t_{h, j} \leq t_{i, p} \quad (A-58)
\end{align*}
\]

\[
\begin{align*}
\text{II- iv - a} & \quad I^h_{i, k} > 0 \implies t_{i, p} - t_{h, k} > 0 \implies t_{i, p} > t_{h, k} \quad (A-59) \\
I^i_{h, k} > 0 & \implies t_{h, p} - t_{i, k} > 0 \implies t_{h, p} > t_{i, k} \quad (A-60)
\end{align*}
\]

\[
\begin{align*}
\text{II- iv - a - 3} & \quad I^h_{i, k+1} = 0 \implies t_{i, p} + t_{i, k} - t_{h, k} - t_{h, k+1} \leq 0 \\
I^i_{h, k+1} = 0 & \implies t_{h, p} + t_{h, k} - t_{i, k} - t_{i, k+1} \leq 0 \\
& \implies T_h < T_i
\end{align*}
\]

\[
\begin{align*}
\text{II- iv - a - 2} & \quad I^h_{i, k+1} > 0 \implies t_{i, p} + t_{i, k} - t_{h, k} - t_{h, k+1} > 0 \quad (A-61) \\
I^i_{h, k+1} = 0 & \implies t_{h, p} + t_{h, k} - t_{i, k} - t_{i, k+1} \leq 0 \quad (A-62)
\end{align*}
\]

(A-61) and (A-62) result in \( t_{h, k} < t_{i, k} \), which contradicts (A-3) => infeasible

\[
\begin{align*}
\text{II- iv - a - 3} & \quad I^h_{i, k+1} = 0 \implies t_{i, p} + t_{i, k} - t_{h, k} - t_{h, k+1} \leq 0 \\
I^i_{h, k+1} > 0 & \implies t_{h, p} + t_{h, k} - t_{i, k} - t_{i, k+1} > 0 \\
& \implies T_h < T_i
\end{align*}
\]

\[
\begin{align*}
\text{II- iv - a - 4} & \quad I^h_{i, k+1} > 0 \implies t_{i, p} + t_{i, k} - t_{h, k} - t_{h, k+1} > 0 \\
I^i_{h, k+1} > 0 & \implies t_{h, p} + t_{h, k} - t_{i, k} + t_{i, k+1} > 0
\end{align*}
\]
\[ T_i - T_h = (t_{i, j} + t_{i, k} - t_{h, j} - t_{h, k}) + 2(t_{h, k} - t_{i, k}) \]

By (A-1) and (A-61), \( T_i - T_h > 0 \)

\[ \Rightarrow T_h < T_i \]

II- iv - b

\[ l^h_{i, k} = 0 \Rightarrow t_{i, p} - t_{h, k} \leq 0 \Rightarrow t_{i, p} \leq t_{h, k} \]

\[ l^i_{h, k} = 0 \Rightarrow t_{h, p} - t_{i, k} \leq 0 \Rightarrow t_{h, p} \leq t_{i, k} \]

II- iv - d - 1

\[ l^h_{i, k+1} = 0 \Rightarrow t_{i, k} - t_{h, k+1} \leq 0 \Rightarrow t_{i, k} \leq t_{h, k+1} \]

\[ l^i_{h, k+1} = 0 \Rightarrow t_{h, k} - t_{i, k+1} \leq 0 \Rightarrow t_{h, k} \leq t_{i, k+1} \]

\[ \Rightarrow T_h < T_i \]

II- iv - d - 2

\[ l^h_{i, k+1} > 0 \Rightarrow t_{i, k} - t_{h, k+1} > 0 \Rightarrow t_{i, k} > t_{h, k+1} \] (A-63)

\[ l^i_{h, k+1} = 0 \Rightarrow t_{h, k} - t_{i, k+1} \leq 0 \Rightarrow t_{h, k} \leq t_{i, k+1} \] (A-64)

(A-63) and (A-64) contradict (A-3) \( \Rightarrow \) infeasible

II- iv - d - 3

\[ l^h_{i, k+1} = 0 \Rightarrow t_{i, k} - t_{h, k+1} \leq 0 \]

\[ l^i_{h, k+1} > 0 \Rightarrow t_{h, k} - t_{i, k+1} > 0 \]

\[ \Rightarrow T_h < T_i \]

II- iv - d - 4

\[ l^h_{i, k+1} > 0 \Rightarrow t_{i, k} - t_{h, k+1} > 0 \]

\[ l^i_{h, k+1} > 0 \Rightarrow t_{h, k} - t_{i, k+1} > 0 \]

\[ \Rightarrow T_i - T_h = (t_{i, j} + t_{i, k} - t_{h, j} - t_{h, k}) + (t_{h, k} - t_{i, k}) \]

By (A-1) and (A-61), \( T_i - T_h > 0 \)

\[ \Rightarrow T_h < T_i \]

II- iv - c

\[ l^h_{i, k} > 0 \Rightarrow t_{i, p} - t_{h, k} > 0 \Rightarrow t_{i, p} > t_{h, k} \] (A-65)
\[ I_{i,h,k}^i = 0 \implies t_{h,p} - t_{i,k} \leq 0 \implies t_{h,p} \leq t_{i,k} \quad (A-66) \]

(A-65) and (A-66) contradict (A-3) => infeasible

\[ \begin{align*}
  & I_{i,h,k}^h = 0 \implies t_{i,p} - t_{h,k} \leq 0 \implies t_{i,p} \leq t_{h,k} \quad (A-67) \\
  & I_{i,h,k}^i > 0 \implies t_{h,p} - t_{i,k} > 0 \implies t_{h,p} > t_{i,k} \quad (A-68)
\end{align*} \]

\[ \begin{align*}
  & I_{i,h,k+1}^h = 0 \implies t_{h,k} - t_{h,k+1} \leq 0 \implies t_{h,k} \leq t_{h,k+1} \\
  & I_{i,h,k+1}^i = 0 \implies t_{h,k} + t_{h,p} - t_{i,k} - t_{i,k+1} \leq 0 \\
  & \implies T_h < T_i
\end{align*} \]

\[ \begin{align*}
  & I_{i,k+1}^h > 0 \implies t_{h,k} - t_{h,k+1} > 0 \implies t_{h,k} > t_{h,k+1} \quad (A-69) \\
  & I_{i,h,k+1}^i = 0 \implies t_{h,k} + t_{h,p} - t_{i,k} - t_{i,k+1} \leq 0 \quad (A-70) \\
  & \text{By (A-69) and (A-70), } t_{h,p} < t_{i,k} \quad (A-71)
\end{align*} \]

(A-71) contradicts (A-68) => infeasible

\[ \begin{align*}
  & I_{i,k+1}^h = 0 \implies t_{h,k} - t_{h,k+1} \leq 0 \implies t_{h,k} \leq t_{h,k+1} \quad (A-72) \\
  & I_{i,h,k+1}^i > 0 \implies t_{h,k} + t_{h,p} - t_{i,k} - t_{i,k+1} > 0 \quad (A-73) \\
  & \text{By (A-68) and (A-73), } t_{h,k} > t_{h,k+1} \text{, which contradicts (A-72) => infeasible}
\end{align*} \]

\[ \begin{align*}
  & I_{i,k+1}^h > 0 \implies t_{h,k} - t_{h,k+1} > 0 \quad (A-74) \\
  & I_{i,h,k+1}^i > 0 \implies t_{h,k} + t_{h,p} - t_{i,k} - t_{i,k+1} > 0 \quad (A-75) \\
  & \text{By (A-68), (A-74) and (A-75), } T_h < T_i
\end{align*} \]

All combinations for (IV) are as follows:
\[ I^h_{i,p} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} - t_{h,p} > 0 \] (A-76)

\[ I^i_{h,p} > 0 \Rightarrow t_{h,e} + t_{h,j} - t_{i,j} - t_{i,p} > 0 \] (A-77)

\[ I^h_{i,k} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} - t_{h,k} > 0 \]

\[ I^i_{h,k} > 0 \Rightarrow t_{h,e} + t_{h,j} - t_{i,j} - t_{i,k} > 0 \]

\[ I^h_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \]

\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,k+1} \leq 0 \]

\[ T_h < T_i \]

\[ I^h_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]

\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,k+1} \leq 0 \]

\[ T_i - T_h = t_{h,k+1} - t_{i,e} \]

If \( t_{h,k+1} > t_{i,e} \) then, \( T_h < T_i \).

\[ I^h_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \] (A-76)

\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,k+1} > 0 \] (A-77)

(A-76) and (A-77) contradicts (A-1)

\[ T_h < T_i \] infeasible

\[ I^h_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]

\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,k+1} > 0 \]

\[ T_i - T_h = t_{h,j} - t_{h,k} - t_{i,j} - t_{i,k} < 0 \]

\[ T_h < T_i \] violation.
Notice that the additivity assumption does not provide an optimal solution.

\[ I^h_{i,k} = 0 \Rightarrow t_{i,e} - t_{h,k} \leq 0 \Rightarrow t_{i,e} \leq t_{h,k} \quad (A-78) \]

\[ l^i_{h,k} = 0 \Rightarrow t_{h,e} + t_{h,j} - t_{i,j} - t_{i,k} \leq 0 \]

**IV - i - b - 1**

\[ I^h_{i,k+1} = 0 \Rightarrow t_{i,k} - t_{h,k+1} \leq 0 \Rightarrow t_{i,k} \leq t_{h,k+1} \]

\[ l^i_{h,k+1} = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \Rightarrow t_{h,k} \leq t_{i,k+1} \]

\[ \Rightarrow T_h < T_i \]

**IV - i - b - 2**

\[ I^h_{i,k+1} > 0 \Rightarrow t_{i,k} - t_{h,k+1} > 0 \Rightarrow t_{i,k} > t_{h,k+1} \quad (A-79) \]

\[ l^i_{h,k+1} = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \Rightarrow t_{h,k} \leq t_{i,k+1} \quad (A-80) \]

(A-79) and (A-80) contradict (A-3)

\[ \Rightarrow \text{infeasible} \]

**IV - i - b - 3**

\[ I^h_{i,k+1} = 0 \Rightarrow t_{i,k} - t_{h,k+1} \leq 0 \Rightarrow t_{i,k} \leq t_{h,k+1} \]

\[ l^i_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \Rightarrow t_{h,k} > t_{i,k+1} \]

\[ \Rightarrow T_h < T_i \]

**IV - i - b - 4**

\[ I^h_{i,k+1} > 0 \Rightarrow t_{i,k} - t_{h,k+1} > 0 \Rightarrow t_{i,k} > t_{h,k+1} \quad (A-81) \]

\[ l^i_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \Rightarrow t_{h,k} > t_{i,k+1} \quad (A-82) \]

By (A-81), (A-82), (A-3) \[ \Rightarrow T_h < T_i \]

**IV - i - c**

\[ I^h_{i,k} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,k} > 0 \]

\[ l^i_{h,k} = 0 \Rightarrow t_{h,e} + t_{h,j} - t_{i,j} - t_{i,k} \leq 0 \]
IV- i - c - 1  \[ I^h_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \Rightarrow t_{i,e} \leq t_{h,k+1} \]

\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \Rightarrow t_{h,k} \leq t_{i,k+1} \Rightarrow T_h < T_i \]

IV- i - c - 2  \[ I^h_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]

\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \Rightarrow T_i - T_h = t_{h,k+1} - t_{i,e} \]

IV- i - c - 3  \[ I^h_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \]

\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \Rightarrow T_h < T_i \]

IV- i - c - 4  \[ I^h_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]

\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \Rightarrow T_i - T_h = t_{h,k} - t_{i,e} < 0 \Rightarrow T_i < T_h \text{ violation.} \]

Notice that the additivity assumption does not provide optimal solution.

IV- i - d  \[ I^h_{i,k} = 0 \Rightarrow t_{i,e} - t_{h,k} \leq 0 \Rightarrow t_{i,e} \leq t_{h,k} \quad (A-83) \]

\[ I^i_{h,k} > 0 \Rightarrow t_{h,e} + t_{h,j} - t_{i,j} - t_{i,k} > 0 \]

(A-83) is in contradiction with (A-2)(A-9)

\[ \Rightarrow \text{infeasible} \]

IV- ii  \[ I^h_{i,p} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,p} > 0 \]
\[ I^i_{h,p} = 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,p} \leq 0 \]

**IV- ii - a**

\[ I^{h}_{i,k} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} - t_{h,k} > 0 \]

\[ I^i_{h,k} > 0 \Rightarrow t_{h,p} - t_{i,k} > 0 \Rightarrow t_{h,p} > t_{i,k} \quad (A-84) \]

**IV- ii - a - 4**

\[ I^{h}_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} \leq 0 \]

\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]

\[ \Rightarrow T_h < T_i \]

**IV- ii - a - 2**

\[ I^{h}_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} > 0 \]

\[ I^i_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]

\[ \Rightarrow T_i - T_h = t_{h,k+1} - t_{i,e} \]

**IV- ii - a - 3**

\[ I^{h}_{i,k+1} = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} \leq 0 \]

\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} > 0 \]

\[ \Rightarrow T_h < T_i \]

**IV- ii - a - 4**

\[ I^{h}_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} + t_{h,k} - t_{h,k+1} > 0 \]

\[ I^i_{h,k+1} > 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} > 0 \]

\[ \Rightarrow T_i - T_h = t_{h,k+1} - t_{i,e} \]

**IV- ii - b**

\[ I^{h}_{i,k} = 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} \leq 0 \Rightarrow t_{i,e} \leq t_{h,j} \quad (A-85) \]

\[ I^i_{h,k} = 0 \Rightarrow t_{h,p} - t_{i,k} \leq 0 \Rightarrow t_{h,p} \leq t_{i,k} \]

\[ (A-84) \text{ contradicts } (A-8) \Rightarrow \text{ infeasible} \]

**IV- ii - c**

\[ I^{h}_{i,k} > 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} > 0 \]

\[ I^i_{h,k} = 0 \Rightarrow t_{h,p} - t_{i,k} \leq 0 \Rightarrow t_{h,p} \leq t_{i,k} \quad (A-86) \]
IV- ii - c - 1 \[ I_{i,k+1}^h = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ I_{h,k+1}^h = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]
\[ \Rightarrow T_h < T_i \]

IV- ii - c - 2 \[ I_{i,k+1}^h > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I_{h,k+1}^h = 0 \Rightarrow t_{h,k} - t_{i,k+1} \leq 0 \]
\[ \Rightarrow T_i - T_h = t_{h,k+1} - t_{i,e} \]

IV- ii - c - 3 \[ I_{i,k+1}^h = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ \Rightarrow t_{i,e} < t_{h,k} \] (A-87)
\[ I_{h,k+1}^h > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \]

(A-87) contradicts (A-9) and (A-2) \[ \Rightarrow \text{infeasible} \]

IV- ii - c - 4 \[ I_{i,k+1}^h > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I_{h,k+1}^h > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \]
\[ \Rightarrow T_i - T_h = t_{h,k} - t_{i,e} \]

IV- ii - d \[ I_{i,k}^h = 0 \Rightarrow t_{i,e} + t_{i,j} - t_{h,j} + t_{h,k} \leq 0 \Rightarrow t_{i,e} \leq t_{h,k} \] (A-88)
\[ I_{h,k}^i > 0 \Rightarrow t_{h,p} - t_{i,k} \leq 0 \]

(A-87) contradicts (A-2) and (A-9) \[ \Rightarrow \text{infeasible} \]

IV- iii \[ I_{i,p}^h = 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,p} > 0\] (A-89)
\[ I_{h,p}^h > 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,p} \leq 0\] (A-90)

(A-89) and (A-90) contradicts (A-1) \[ \Rightarrow \text{infeasible} \]
IV- iv 
\[ l^h_{i,p} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,p} > 0 \]
\[ l^i_{h,p} = 0 \Rightarrow t_{h,e} + t_{h,j} + t_{h,k} - t_{i,j} - t_{i,k} - t_{i,p} \leq 0 \]

IV- iv - a
\[ l^h_{i,k} > 0 \Rightarrow t_{i,p} - t_{h,k} > 0 \Rightarrow t_{i,p} > t_{h,k} \]
\[ l^i_{h,k} > 0 \Rightarrow t_{h,p} - t_{i,k} > 0 \Rightarrow t_{h,p} > t_{i,k} \]

IV- iv - a - 1
\[ l^h_{i,k+1} = 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \Rightarrow t_{i,p} \leq t_{h,k+1} \]
\[ l^i_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \Rightarrow t_{h,p} \leq t_{i,k+1} \]
\[ \Rightarrow T_h < T_i \]

IV- iv - a - 2
\[ l^h_{i,k+1} > 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \]  \( \text{(A-91)} \)
\[ l^i_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]
\[ \Rightarrow t_{h,p} \leq t_{i,k+1} \]  \( \text{(A-92)} \)
\[ \text{(A-91) and (A-92) contradict (A-3)} \]
\[ \Rightarrow \text{infeasible} \]

IV- iv - a - 3
\[ l^h_{i,k+1} = 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \Rightarrow t_{i,p} \leq t_{h,k+1} \]
\[ l^i_{h,k+1} > 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0 \]
\[ \Rightarrow T_h < T_i \]

IV- iv - a - 4
\[ l^h_{i,k+1} > 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \]
\[ l^i_{h,k+1} > 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} > 0 \]
\[ T_i - T_h = t_{i,j} + t_{i,k} + t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} - t_{h,k} - t_{h,j} + t_{i,p} \]
\[ + t_{i,k} - t_{h,k} - t_{h,k+1} \]
\[ \Rightarrow T_h < T_i \]
IV- iv - b \[ I^h_{i,k} = 0 \Rightarrow t_{i,p} - t_{h,k} \leq 0 \Rightarrow t_{i,p} \leq t_{h,k} \]
\[ I^l_{h,k} = 0 \Rightarrow t_{h,p} - t_{i,k} \leq 0 \Rightarrow t_{h,p} \leq t_{i,k} \]

IV- iv - b - 1 \[ I^h_{i,k+1} = 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ I^l_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]
\[ \Rightarrow t_{h,p} < t_{i,k+1} \]
\[ \Rightarrow T_h < T_i \]

IV- iv - b - 2 \[ I^h_{i,k+1} > 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} > 0 \quad (A-93) \]
\[ I^l_{h,k+1} = 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \quad (A-94) \]
(A-93) and (A-94) contradicts (A-92)
\[ \Rightarrow \text{infeasible} \]

IV- iv - b - 3 \[ I^h_{i,k+1} = 0 \Rightarrow t_{i,p} + t_{i,k} - t_{h,k} - t_{h,k+1} \leq 0 \]
\[ I^l_{h,k+1} > 0 \Rightarrow t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0 \]
\[ \Rightarrow T_h < T_i \]

IV- iv - b - 4 \[ I^h_{i,k+1} > 0 \Rightarrow t_{i,e} + t_{i,j} + t_{i,k} - t_{h,j} - t_{h,k} - t_{h,k+1} > 0 \]
\[ I^l_{h,k+1} > 0 \Rightarrow t_{h,k} - t_{i,k+1} > 0 \]
\[ T_i - T_h = t_{i,j} + t_{i,k} + t_{h,k} - t_{i,k+1} - t_{h,k} - t_{h,j} - t_{i,e} - t_{i,j} - t_{i,k} \]
\[ + t_{h,j} + t_{h,k} + t_{h,k+1} \]
\[ \Rightarrow T_h < T_i \]

IV- iv - c \[ I^h_{i,k} > 0 \Rightarrow t_{i,p} - t_{h,k} > 0 \Rightarrow t_{i,p} > t_{h,k} \quad (A-95) \]
\[ I^l_{h,k} = 0 \Rightarrow t_{h,p} - t_{i,k} \leq 0 \Rightarrow t_{h,p} \leq t_{i,k} \quad (A-96) \]
(A-95) and (A-96) contradict (A-3) \[ \Rightarrow \text{infeasible} \]
IV- iv - d  

\[ I^h_{i,k} = 0 \implies t_{i,p} - t_{h,k} \leq 0 \implies t_{i,p} \leq t_{h,k} \quad (A-97) \]
\[ I^i_{h,k} > 0 \implies t_{h,p} - t_{i,k} > 0 \implies t_{h,p} > t_{i,k} \quad (A-98) \]

IV- iv - d - 1  

\[ I^h_{i,k+1} = 0 \implies t_{i,k} - t_{h,k+1} \leq 0 \]
\[ I^i_{h,k+1} = 0 \implies t_{h,p} + t_{h,k} - t_{i,k} + t_{i,k+1} \leq 0 \]
\[ \implies t_{h,p} \leq t_{i,k+1} \]
\[ \implies T_h < T_i \]

IV- iv - d - 2  

\[ I^h_{i,k+1} > 0 \implies t_{i,k} - t_{h,k+1} > 0 \implies t_{i,k} > t_{h,k+1} \quad (A-99) \]
\[ I^i_{h,k+1} = 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} \leq 0 \quad (A-100) \]
\[ \implies t_{h,p} < t_{i,k+1} \quad (A-101) \]

By (A-3) and (A-98), \( t_{h,k} < t_{i,k+1} \), which contradicts (A-99) \implies \text{infeasible} \]

IV- iv - d - 3  

\[ I^h_{i,k+1} = 0 \implies t_{i,k} - t_{h,k+1} \leq 0 \implies t_{h,k} \leq t_{h,k+1} \]
\[ I^i_{h,k+1} > 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0 \]
\[ \implies T_h < T_i \]

IV- iv - d - 4  

\[ I^h_{i,k+1} > 0 \implies t_{i,k} - t_{h,k+1} > 0 \implies t_{i,k} > t_{h,k+1} \quad (A-99) \]
\[ I^i_{h,k+1} > 0 \implies t_{h,p} + t_{h,k} - t_{i,k} - t_{i,k+1} > 0 \]
\[ T_i - T_h = t_{i,j} + t_{i,k} + t_{h,k} - t_{i,k} - t_{i,k+1} - t_{h,k} - t_{h,j} - t_{i,k} + t_{h,k+1} \]
\[ T_i - T_h = t_{i,j} + t_{i,k} + t_{h,p} - t_{i,k} - t_{h,j} - t_{i,k} \]
\[ \implies T_h < T_i \]
APPENDIX 3

LIST OF NOTATION

\( \alpha \) The parameter used for scaling weights
\( c_{i,j} \) Completion time of job i on machine j
\( C_{k,h} \) The Objective value for the solution space \( s_{k,h} \)
\( t_{i,j} \) Processing time of job j on machine j
\( f_{k,i,j} \) Parameter function for job j on machine j, given objective function \( g_k \)
\( F_{i,j} \) Flow time of job i on machine j
\( g_k \) Performance objective function k
\( I_{i,j} \) Idle time of machine j to process job i
\( J_i \) Job i
\( m \) The number of the machines
\( MAV \) Multiattribute value
\( n \) The number of jobs
\( N^a(s_{k,h}) \) The neighborhood around the solution \( s_{k,h} \) with the transform a
\( r_i \) Ready time
\( R^{(superscript)} \) The superscript indicates "reverse"
\( s_i \) Sequence i
\( s_{k,i} \) Sequence i given performance objective \( g_k \)
$S_k$  Solution space for the performance objective $g_k$

$t_{i,j}$  Processing time of job $i$ on machine $j$

$\mathbf{t}^{**}$  $t^{**} = t_{i,j}, \ i=1, \ldots, n ; j=1, \ldots, m$

$V_{k,j}(.)$  Unidimensional value function for machine $j$, given $g_k$

$V_k(\ldots, \ldots)$  Multidimensional value function for machine $j$, given $g_k$

$g_k$  

$w_j$  Scaling weight for $U_{k,j}(\ldots)$ where is $g_k$ makespan

$W_{i,j}$  Waiting time of job $i$ for machine $j$

$X$  Parameter space