Research on Scheduling Problems in Machining and Assembly Processes using Multiobjective Evolutionary Algorithms

ZHANG, Wenqiang

July 2011
Research on Scheduling Problems in Machining and Assembly Processes using Multiobjective Evolutionary Algorithms

ZHANG, Wenqiang

Graduate School of Information, Production and Systems
Waseda University

July 2011
# Contents

1 Introduction
   1.1 Research Background and Motivation ........................................ 1
   1.2 Assembly Line Balancing Problems ........................................... 3
   1.3 Process Planning and Scheduling Problems .................................... 9
   1.4 Research Objectives .................................................................. 18
   1.5 Methodology and Limitations ...................................................... 19
   1.6 Outline of the Dissertation ......................................................... 20

2 Multiobjective Evolutionary Algorithms ......................................... 23
   2.1 Multiobjective Problems .......................................................... 23
   2.2 Multiobjective Evolutionary Algorithms ..................................... 25

3 Multiobjective smALB Considering Workers’ Capabilities ....................... 37
   3.1 Introduction ............................................................................ 37
   3.2 Single-model Assembly Line Balancing Problem ............................. 39
   3.3 Mathematical Formulation ......................................................... 41
   3.4 Multiobjective Genetic Algorithm Approach .................................. 42
   3.5 Experiments and Discussion ....................................................... 54
   3.6 Summary .................................................................................. 58

4 Multiobjective mALB with Novel Cycle Time Reference ......................... 59
   4.1 Introduction ............................................................................ 59
   4.2 Mixed-model Assembly Line Balancing Problem ............................ 60
   4.3 Mathematical Formulation ......................................................... 64
   4.4 Multiobjective Genetic Algorithm Approach .................................. 67
   4.5 Experiments and Discussion ........................................................ 74
   4.6 Summary .................................................................................. 80

5 Fast and Effective Multiobjective Evolutionary Algorithm for PPS ........... 81
   5.1 Introduction ............................................................................ 81
   5.2 Multiobjective PPS Problem ....................................................... 83
   5.3 Fast and Effective Multiobjective Evolutionary Algorithm ............... 86
   5.4 Experiments and Discussion ....................................................... 93
   5.5 Summary .................................................................................. 98

6 Fast Multiobjective Genetic Algorithm with Archive for PPS in DMS ........ 101
   6.1 Introduction ............................................................................ 101
   6.2 Process Planning and Scheduling Problem in Distributed Environments 103
   6.3 Mathematical Formulation ......................................................... 104
6.4 Fast Multiobjective Genetic Algorithm with Archive Approach ........ 108
6.5 Experiments and Discussion ............................................. 114
6.6 Summary .............................................................................. 119

7 Conclusions .............................................................................. 121
  7.1 Multiobjective smALB Problem Considering Workers’ Capabilities .... 121
  7.2 Multiobjective mALB Problem with Novel Cycle Time Reference .... 122
  7.3 Multiobjective PPS Problem ................................................. 123
  7.4 Multiobjective PPS Problem in DMS .................................... 124
  7.5 Future Research ............................................................... 125

Acknowledgments ....................................................................... 127

List of Publications .................................................................... 129

A Appendices to Chapters ......................................................... 131

List of Figures ............................................................................ 140
List of Tables ............................................................................. 143

Bibliography ............................................................................... 147

Index ......................................................................................... 154
Chapter 1

Introduction

1.1 Research Background and Motivation

The broadening global financial crisis has hit much of the global economy especially in manufacturing and industrial process fields. How to improve the production efficiency and reduce the production cost become urgent matter for processes industries.

Scheduling, as a decision-making process, plays an important role in most manufacturing system (MS), transportation and service industries as well as in most information processing environments. The primary aim of this study is to present an investigation on multiobjective evolutionary algorithms (moEAs) for two real-world scheduling problems in machining and assembly processes, assembly line balancing (ALB) and Process planning and scheduling (PPS), which are important and intractable combinatorial optimization problems in MS.

ALB and PPS are the use of manufacturing resources, such as machines and tools, as well as labor to make products for use or sale. In PPS, raw materials are often to be processed into final products, which may be used for manufacturing other, more complex, products, or sold to end users by logistics transportation system. In ALB, the above products are added to a product in a sequential manner to create a finished product, such as a household appliance or automobile.

ALB is used to balance workloads on an assembly line by assigning numbers of workers or machines to each task. PPS deals with resource (machines, tools) utilization and time allocation of manufacturing operations of various parts effectively and inexpensively. Much investigation and study have proven that ALB and PPS enable the MS to achieve better product quality, higher productivity, and lower production cost. Therefore, solving ALB and PPS problems has significant economic and social consequences in today’s productivity-oriented world.

PPS and ALB are used to generate an effective optimal solution (schedule), which tells a production facility what to make, when, with which staff, and on which equipment, to process the products effectively. Unfortunately, determination of such a solution is a difficult combinatorial optimization problem. Seeking an optimal solution rapidly and effectively from all of the permutations, combinations of all of the tasks, manufacturing resources and human resources according to specified criteria is very intractable.

In modern industry, an assembly line (AL) is a manufacturing process in which interchangeable parts are added to a product in a sequential manner to create a finished product. ALB problem is a difficult combinatorial optimization problems arising fre-
quent in manufacturing, which deals with the allocation of the tasks among stations satisfying the precedence relationship constraint between tasks, so as to some given objective functions such as cycle time, line efficiency, production cost, stations workload, etc., are optimized. Because processing tasks may require widely different task times, it is difficult to assign task times to stations equally.

With these increasing and varied demands of customers, manufacturers are forced to produce different models of the same product on an AL so that the potential market is not lost. In such environments, mixed-model assembly lines (mAL) appear to be the most appropriate. In a mAL, a set of similar models of a product, e.g., versions of a car, which may differ from each other with respect to size, color etc., can be assembled simultaneously. Therefore, mixed-model assembly line balancing (mALB) problem becomes more difficult.

In MS, PPS is used to process a set of parts with operations on machines by selecting suitable manufacturing resources and sequence the operations in order to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved. It is used to determine the most appropriate moment to execute each operation for the launched production orders, taking into account the due date of these orders, a minimum makespan, a balanced resource utilization, etc., to obtain high productivity in the workshop [1], [2].

In the recent years, manufacturing is gradually changing to a distributed environment with increasing dynamism. In order to increase their international competitiveness, locally or globally, customer satisfaction is treated with priority. This leads to mass customization and more complex manufacturing processes, from shop floors to every level along manufacturing supply chains. Different capabilities of factories, different production efficiency and various constraints make the PPS problem of distributed manufacturing system (DMS) more and more complex.

In solving ALB and PPS problems, most of previous researchers concentrated on optimization of one single objective. Unfortunately, in the most practical situations, the PPS and ALB often encounter an array of conflicting criteria. In any case, PPS and ALB are looking for a solution for which each objective has been optimized to the extent that if they try to improve it any further, then the other objective(s) will be exacerbated as a result. It has led to multiobjective PPS and ALB problems.

As a trade-off between two or more conflicting objectives, the multiobjective problems generate many solutions, named as nondominated solutions or Pareto optimal solutions, with incommensurable quality for decision makers. Decision makers like the multiple solutions with same good quality so that they could select from them according to their subjective preferences.

In the most single objective optimization problem, more problem-dependent considerations are concerned and one global optimal solution and less computation time are desired. However, multiobjective optimization problem (moOP) needs additional cares in convergence mechanism of finding sufficient number of global Pareto optimal solutions as soon as possible, and dispersion mechanism of distributing them as evenly as possible. Moreover, reduction of computation time becomes more difficult accordingly. The above special concerns make ALB and PPS extremely complicated in nature so that optimization of them becomes a big challenge.

The following sections will introduce the related works of this research.
1.2 Assembly Line Balancing Problems

AL is flow-oriented MS which is typical in the modern industrial production. Henry Ford invented the concept of AL for automobile manufacturing in 1913. He was the first to introduce a moving belt in a factory, where the workers were able to build the famous model-T cars, one piece at a time instead of one car at a time. Since then, the AL concept revolutionized the way of the products was made while reducing the cost of production. Over the years, the design of efficient AL received considerable attention of both companies and academicians. A well-known assembly design problem is the assembly line balancing (ALB), which deals with the allocation of the tasks among stations (workstations) so that a given objective function is optimized.

1.2.1 Problem Description

An AL contains a sequence of stations arranged along a conveyor belt or a similar material handling system. The work pieces are consecutively launched down the line and are moved from station to station by conveyor belt. Stations are used to assemble components into a final product. At each station, certain tasks are repeatedly processed according to the cycle time (maximum or average time available for each work cycle). Manufacturing a product on an AL requires assign tasks to stations as equally as possible, that is ALB.

Balancing AL is a difficult combinatorial optimization problem arising frequently in manufacturing, for its precedence relationship constraint between tasks. Usually, the number of stations, cycle time, workload balance, and other factors, are used to measure the performance of an AL. Balancing AL is a crucial task for manufacturing companies in order to improve productivity and minimize production costs. Because processing tasks may require widely different times, it is difficult to assign task times to stations equally. This leads to idle time at stations. One of the classical objectives of ALB problems is to minimize this idle time.

Table 1.1 presents the data set for an example of ALB, which contains 10 tasks. Using this data set, the precedence relationship graph is constructed as shown in Fig. 1.1.

Table 1.1: Data set of an example for ALB

<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessor Proj()</th>
<th>Processing time (time unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ø</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>Ø</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>Ø</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>{2,3}</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>{4}</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>Ø</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>{6}</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>Ø</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>{7,8,9}</td>
<td>26</td>
</tr>
</tbody>
</table>
The precedence graph contains 10 nodes for tasks (node weights for task processing times and arcs for orderings). For example, the processing time for task 6 is 28 time units. For the processing of task 6, task 4 (direct predecessors) and task 1 (indirect predecessors) must be completed. Likewise, task 6 must be completed before tasks 8 (direct successors), and tasks 10 (indirect successors) start processing.

Suppose there are \( n \) tasks needed to be processed in \( m \) stations. The notations for describing this problem can be summarized as follows.

**Indices**

\( i \): index of the stations \((i = 1, \ldots, m)\).

\( j, k \): index of the tasks \((j, k = 1, \ldots, n)\).

**Parameters**

\( m \): number of stations actually employed.

\( M \): maximum number of stations available \((m \leq M)\).

\( n \): number of tasks.

\( c_T \): cycle time of the assembly line.

\( t_j \): processing time of task \( j \).

\( S_i \): the set of tasks assigned to station \( i \).

\( T_s \): a task sequence.

\( \text{Suc}(j) \): the set of direct successors of task \( j \).

\( \text{Pre}(j) \): the set of direct predecessors of task \( j \).

\( u_i \): utilization of the station \( i \). \( u_i = \max_{1 \leq i \leq m} \left\{ t(S_i) \right\} / t(S_i) \)

\( \bar{u} \): average utilization of total stations. \( \bar{u} = \frac{1}{m} \sum_{i=1}^{m} u_i \)

**Decision variables**

\[
    x_{ij} = \begin{cases} 
        1, & \text{if task } j \text{ is assigned to station } i \\
        0, & \text{otherwise} 
    \end{cases}
\]
Mathematical Model

The mathematical model for the ALB can be stated as follows:

\begin{align*}
\text{minimize} \quad & \{c_T\} \\
\text{where} \quad & c_T = \max_{1 \leq i \leq M} \left\{ \sum_{j=1}^{n} t_j x_{ij} \right\} \\
\text{s.t.} \quad & \sum_{i=1}^{M} x_{ij} = 1, \quad \forall \ j \\
& \sum_{i=1}^{M} i x_{ik} \leq \sum_{i=1}^{M} i x_{ij}, \quad \forall \ j, \forall \ k \in \text{Pre}(j) \\
& t(S_j) = \sum_{j \in S_i} t_j = \sum_{j=1}^{n} t_j x_{ij} \leq c_T, \quad \forall \ i \\
& x_{ij} = 0 \text{ or } 1, \quad \forall \ i, j
\end{align*} \tag{1.1-1.5}

In this mathematical model, the first objective (1.1) of the model is to minimize the cycle time. The constraints given in equations (1.2)-(1.5) are used to formulate the general feasibility of the problem. The constraint given in equation (1.2) states that each task must be assign to one and only one station. Inequity (1.3) represents the precedence constraints and it states that the direct predecessor of task \(j\) must be assign to a station, which is in front of or the same as the station that task \(j\) is assigned in. This constraint stresses that if a task is assigned to a station, then the predecessor of this task must be already assigned to a station. Inequity (1.4) denotes that the available time at each station should be less than or equal to the given cycle time. Constraint given in equation (1.5) represents the usual integrity restriction.

1.2.2 Classification of ALB

Number of products

Based on the number of products, ALB problem can be classified into three types [3]:

- **single-model assembly line balancing** (smALB)
- **multi-model assembly line balancing** (muALB)
- **mixed-model assembly line balancing** (mALB)

The smALB manufactures one homogeneous product continuously in large quantities (mass production) (see Fig. 1.2).

![Figure 1.2: Single-model assembly line](image)

The muALB model involves more than one product produced in batches (see Fig. 1.3).
6 Introduction

Figure 1.3: Multi-model assembly line

The mALB refers to assembly lines which are capable of producing a variety of similar product models simultaneously and continuously (not in batches) (see Fig. 1.4).

Figure 1.4: Mixed-model assembly line

Model structure

Based on the model structure, ALB models can be classified into two parts:

- *simple assembly line balancing* (sALB)
- *general assembly line balancing* (gALB)

The sALB, the simplest version of the ALB model and the special version of smALB model, involves production of only one product with features such as paced line with fixed cycle time, deterministic independent processing times, no assignment restrictions, serial layout, one sided stations, equally equipped stations and fixed rate launching. The gALB model includes all of the models that are not sALB, such as balancing of mixed-model, parallel, u-shaped and two sided lines with stochastic dependent processing times; thereby more realistic ALB models can be formulated by gALB [4].

Layout of production system

According to the layout of the production system, ALB can be classified into three types mainly:

- *Serial assembly line*
- *U-shaped assembly line*
- *Parallelism assembly line*

The serial assembly line is the traditional layout of assembly line which single stations are arranged in a straight line along a conveyor belt (see Fig. 1.5)

The stations are arranged within a narrow U in the U-shaped assembly line. Workers are allowed to work on either side of the U consequently (see Fig. 1.6).

In parallelism assembly line, the duplication of stations is used as identical stations which execute the same tasks with the same equipment (see Fig. 1.7).
Objectives

Additionally, several versions of ALB problems arise by varying the objective function [3].

- **Type-F** is an objective independent problem, which is to establish whether or not a feasible line balance exists.
- **Type-1** tries to minimize the number of stations for a given cycle time.
- **Type-2** tries to minimize the cycle time for a given number of stations.
- **Type-E** is the most general problem version, which tries to maximize the line efficiency by simultaneously minimizing the cycle time and a number of stations.
- **Type-3** corresponds to maximization of workload smoothness.
- **Type-4** corresponds to maximization of work relatedness.
- **Type-5** corresponds to multiple objectives with Type-3 and Type-4 [5].

1.2.3 Cutting-Edge of the Research Survey

Since the ALB model was first described by Helgeson et al. [6], many solution approaches have been proposed. The complexity of the ALB renders optimum seeking methods impractical for instances of more than a few tasks and/or stations. If there are $m$ tasks and $r$ preference constraints, then there are $m!/2^r$ possible task sequences [7].
Therefore, it can be time consuming for optimum seeking methods to obtain an optimal solution within this vast search space \cite{8}. Several optimum seeking methods, such as linear programming \cite{9}, integer programming \cite{10}, dynamic programming \cite{11} and branch-and-bound approaches \cite{12} have been employed to resolve ALB. However, none of these methods has proven to be of practical use for large problems due to their computational inefficiency.

Due to ALB model falls into the NP-hard class of combinatorial optimization problems \cite{13}, in recent years, to provide an alternative to traditional optimization techniques, numerous research efforts have been directed towards the development of heuristics \cite{14} and meta-heuristics. While heuristic methods generating one or more feasible solutions were mostly developed until mid 90s; meta-heuristics such as tabu search \cite{15}, simulated annealing \cite{16}, genetic algorithms \cite{17}, ant colony optimization \cite{18}, and particle swarm algorithm \cite{19} have been the focus of researchers in the last decade.

For more information can be refereed the following studies. Baybars \cite{7} that surveys the exact (optimal) methods. Talbot et al. \cite{20} that compare and evaluate the heuristic methods developed. Ghosh and Gagnon \cite{21} that present a comprehensive review and analysis of the different methods for design, balancing and scheduling of assembly systems. Erel and Sarin \cite{22} that present a comprehensive review of the procedures for smALB, muALB and mALB models. Rekiek et al. \cite{23} that focus on optimization methods for the line balancing and resource planning steps of assembly line design. Scholl and Becker \cite{24} that present a review and analysis of exact and heuristic solution procedures for sALB. Becker and Scholl \cite{25} that present a survey on problems and methods for gALB. Rekiek and Delchambre \cite{26} that focus on solutions methods for solving sALB. Ozmehmet Tasan and Tunali \cite{8} that present a comprehensive review of GAs approaches used for solving various ALB models.

Among the meta-heuristics, the application of genetic algorithms (GAs) received a considerable attention from the researchers, since it provides an alternative to traditional optimization techniques by using directed random searches to locate optimum solutions in complex landscapes and it is proven effective in various combinatorial optimization problems. GAs are powerful and broadly applicable stochastic search and optimization techniques based on principles from evolutionary theory \cite{27}.

Falkenauer and Delchambre \cite{17} were the first to solve ALB with GAs. Following them, application of GAs for solving ALB model was studied by many researchers, e.g., \cite{5,28,29}. However, most of the researchers focused on the simplest version of the problem, with single objective and ignored the recent trends, i.e., mixed-model production, u-shaped lines, robotic lines etc., in the complex assembly environments, where ALB models are multiobjective in nature \cite{8}.

### 1.2.4 Insufficiencies of Previous Researches

In assembly processes, as above mentioned, ALB deals with the even allocation of the proper tasks and workers among the proper stations. As one type of ALB, single-model ALB (smALB) concerns with smooth allocation of tasks of one homogeneous product, which is continuously manufactured in large quantities (mass production). The other important one is mixed-model ALB (mALB), which processes several models (versions) of a basic product on the same line. It has been proven so difficult to partition the tasks (of each model) among the stations uniformly. Moreover, consideration of different workers’ capabilities can significantly increase the difficulty of allocation
of tasks and workers into stations evenly. Therefore, optimization of multiobjective smALB and mALB considering workers’ capabilities becomes an intractability problem in MS. Most of researches frequently focused on reducing the cycle time, balancing the station workload, improving the line efficiency and so on.

**For smALB problem**

- There is a lack of papers which have considered arranging the workers according to workers’ capabilities.
- Consideration of single objective problem is more than multiple objectives problem.

**For mALB problem**

- The previous researchers do not consider that the cycle time needs to be associated with the demand ratio of each model in mALB problem.
- There is a lack of papers which have considered arranging the workers according to workers’ capabilities.
- Consideration of single objective problem is more than multiple objectives problem.

### 1.3 Process Planning and Scheduling Problems

#### 1.3.1 Process Planning Problem

The PP is a task that determines the detailed manufacturing requirements for processing a raw material into a completed part, within the available machining resources [30]. The process plan is generally described by operations, machine tools, cutting tools, fixtures, machining parameters, etc. Process planning is the determination of operations (machines, tools, and tool access directions) and their operation sequences for manufacturing a part effectively and economically. All the manufacturing resources are assumed as available in this phase.
In PP, a part is generally described by features such as holes, slots, and bosses. Given a part and a set of manufacturing resources, the PP problem can be defined as follows.

**Operation sequencing** Determine the sequence of executing all operations required for the part so that the precedence relationships among all the operations are maintained.

**Process selection** For each feature, determine one or several operations requirements. This includes the selection of machines, tools, and tool access directions (TAD) based on the feature geometry and available machining resources.

A prismatic part shown in Fig. 1.8(31) was used to introduce PP problem. RS1–RS6 represents the envelope surfaces of the raw stock in Fig. 1.8.

The precedence relationships of the part are shown in Fig. 1.9 where column 2 shows the operations that must be carried out when the operation in the first column has been completed.

![Figure 1.9: Precedence relationship of PP problem with 7 operations](image)

The machining resources are shown in Table. 1.2

<table>
<thead>
<tr>
<th>Feature-ID</th>
<th>Operation-ID</th>
<th>Operations</th>
<th>TAD candidates</th>
<th>Machine candidates</th>
<th>Tool candidates</th>
<th>Machining time (time unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>op1</td>
<td>Milling</td>
<td>- y</td>
<td>m2, m3</td>
<td>t6, t9</td>
<td>12, 15</td>
</tr>
<tr>
<td>f2</td>
<td>op2</td>
<td>Milling</td>
<td>- y</td>
<td>m5, m3</td>
<td>t9, t10</td>
<td>21, 18</td>
</tr>
<tr>
<td>f3</td>
<td>op3</td>
<td>Milling</td>
<td>- z</td>
<td>m2, m3</td>
<td>t3</td>
<td>18, 25</td>
</tr>
<tr>
<td>f4</td>
<td>op4</td>
<td>Milling</td>
<td>+ x, - x</td>
<td>m2, m3</td>
<td>t1, t3</td>
<td>27, 25</td>
</tr>
<tr>
<td>f5</td>
<td>op5</td>
<td>Drilling</td>
<td>+ x</td>
<td>m1, m2, m3</td>
<td>t2</td>
<td>20, 18</td>
</tr>
<tr>
<td>f6</td>
<td>op6</td>
<td>Milling</td>
<td>- x</td>
<td>m5, m3</td>
<td>t3</td>
<td>14, 15</td>
</tr>
<tr>
<td>f7</td>
<td>op7</td>
<td>Drilling</td>
<td>+ x, + x</td>
<td>m1, m2, m3</td>
<td>t2</td>
<td>20, 11, 18</td>
</tr>
</tbody>
</table>

Fig. 1.8 shows a part composed of 7 features. Each feature can be processed by one or more machining operations (7 operations in total for the part). Each operation can be executed by several alternative plans if different machines, cutting tools or set-up plans are chosen for this operation [32], [33]. A *set-up* is usually defined as a group of operations that are machined on a specified machine with the same fixture. Here, a set-up is equivalently defined as a group of operations with the same tool approach direction (TAD) machined on a machine [34]. For example, a through hole with two
TADs is considered related to two set-ups. A process plan for a part consists of all the operations needed to machine the part and their relevant machines, cutting tools, TADs, and operation sequences. A good process plan of a part is built up based on two elements: (1) the optimized selection of the machine, cutting tool and TAD for each operation, and (2) the optimized sequence of the operations of the part [34].

The PP problem is expressed in the following notation.

**Notations**

**Indices**

- \( j, h \): index of operation, \((j, h = 1, 2, \ldots, J)\).
- \( m \): index of machine, \((m = 1, 2, \ldots, M)\).
- \( l \): index of tool, \((l = 1, 2, \ldots, L)\).
- \( d \): index of tool access direction, \((d = 1, 2, \ldots, D)\).

**Decision variables**

- \( x_{mj}^M \) = \begin{cases} 1, & \text{if operation } o_j \text{ is performed by machine } m_m, \\ 0, & \text{otherwise} \end{cases} 
- \( x_{lj}^T \) = \begin{cases} 1, & \text{if operation } o_j \text{ is performed by tool } t_l, \\ 0, & \text{otherwise} \end{cases} 
- \( x_{dj}^D \) = \begin{cases} 1, & \text{if operation } o_j \text{ is performed by TAD } a_d, \\ 0, & \text{otherwise} \end{cases} 
- \( y_{jh} \) = \begin{cases} 1, & \text{if operation } o_j \text{ is performed directly before operation } o_h, \\ 0, & \text{otherwise} \end{cases} 
- \( \Omega (X, Y) = \begin{cases} 1, & \text{if } X \neq Y, \\ 0, & \text{if } X = Y \end{cases} \)

**Parameters**

- \( J \): number of operations.
- \( M \): number of machines.
- \( L \): number of tools.
- \( D \): number of tool access directions.
- \( O \): set of operations, i.e., \( O = \{ o_j \mid j = 1, 2, \ldots, J \} \).
- \( o_j \): the \( j \)-th operation.
- \( m_m \): the \( m \)-th machine.
- \( t_l \): the \( l \)-th tool.
- \( a_d \): the \( d \)-th tool access direction.
- \( M_j \): set of machines that can process operation \( j \).
- \( A_m \): set of operations that can be processed on machine \( m \).
- \( S \): operation sequence \( S = (s_1, s_2, \ldots, s_k, \ldots, s_f) \) where \( s_k = o_j, j = 1, 2, \ldots, J \).
- \( r_{jh} \): precedence constraints. \( r_{jh} = 1 \), if \( o_j \) is predecessor of \( o_h \); \( 0 \), otherwise.
- \( t_{mj}^M \): machining time of operation \( j \) by machine \( m \).
\[ t^{TM} = \sum_{j=1}^{l} \sum_{m=1}^{M} t_{mj}^M x_{mj} \] (1.6)

\[ t^{MCI} \]: machine change time index. It is same for each machine change.

\[ t_{mj}^M \]: machine change time of operation \( j \) by machine \( m \). It is needed when two adjacent operations \( ((o_{h'}, o_j) \in \{(o_{h'}, o_j) \mid (o_{h'} = s_{w-1}) \land (o_j = s_w)\}) \) are performed on different machines.

\[ t_{mj}^{MCI} = t^{MCI}, j = \{ j' \mid o_j = s_1 \} \]  \hspace{1cm} \text{or}  \hspace{1cm} t_{mj}^{MCI} = t^{MCI} \times \Omega \left( \left\{ m_1 \mid x_{mj}^M = 1 \right\}, \left\{ m_2 \mid x_{mj}^M = 1 \right\} \right), \] (1.7)

\[ t^{TCI} \]: tool change time index. It is same for each tool change.

\[ t_{mj}^{TCI} = t^{TCI}, j = \{ j' \mid o_j = s_1 \} \]  \hspace{1cm} \text{or}  \hspace{1cm} t_{mj}^{TCI} = t^{TCI} \times \left( 1 - \Omega \left( \left\{ m_1 \mid x_{mj}^M = 1 \right\}, \left\{ m_2 \mid x_{mj}^M = 1 \right\} \right) \right), \] (1.9)

\[ t^{TTC} \]: total tool change time.

\[ t_{mj}^{TTC} = t^{TTC} + t^{TCI} \times \sum_{j \in \{ j' \mid o_j = s_1 \}} \Omega \left( \left\{ m_1 \mid x_{mj}^M = 1 \right\}, \left\{ m_2 \mid x_{mj}^M = 1 \right\} \right) \] \hspace{1cm} (1.10)

\[ t^{SCI} \]: set-up change time index. It is same for each set-up change.

\[ t_{mj}^{SCI} = t^{SCI}, j = \{ j' \mid o_j = s_1 \} \]  \hspace{1cm} \text{or}  \hspace{1cm} t_{mj}^{SCI} = t^{SCI} \times \left( 1 - \Omega \left( \left\{ m_1 \mid x_{mj}^M = 1 \right\}, \left\{ m_2 \mid x_{mj}^M = 1 \right\} \right) \right), \] (1.11)
\(t^{\text{TSC}}\): total set-up change time.

\[
t^{\text{TSC}} = t^{\text{SCI}} + t^{\text{SCI}} \times \sum_{j \notin \{(h', f') \mid o_{h'} = s_{k-1} \land o_f = s_k, k = 2, \ldots, J\}} \left( \left( 1 - \Omega \left( \left\{ m_1 \left| x^{M}_{m_1} = 1 \right\} \cup \left\{ m_2 \left| x^{M}_{m_2} = 1 \right\} \right\} \right) \right) \times \Omega \left( \left\{ d_1 \left| x^T_{d_1} = 1 \right\} \cup \left\{ d_2 \left| x^T_{d_2} = 1 \right\} \right\} \right) \right)
\] (1.12)

\(t^{\text{PRE}}_{mj}\): preparation time of operation \(j\) by machine \(m\). The preparation time for an operation consists of the machine change time, the tool change time and the set-up time for the operation. It is necessary to mention that the above three change times are required for the first operation.

\[
t^{\text{PRE}}_{mj} = t^{\text{MC}}_{mj} + t^{\text{TC}}_{mj} + t^{\text{SC}}_{mj}
\] (1.13)

\(t^P_{mj}\): processing time of operation \(j\) by machine \(m\). The preparation time for an operation consists of the preparation time and the machining time for the operation.

\[
t^P_{mj} = t^{\text{PRE}}_{mj} + t^M_{mj}
\] (1.14)

\(t^C_{mj}\): completion time of operation \(j\) by machine \(m\).

**Mathematical Model**

The mathematical model for the PP can be stated as follows:

\[
\text{minimize} \quad \{ t_P \}
\]

where

\[
\{ t_P = t^M + t^{\text{TMC}} + t^{\text{TTC}} + t^{\text{TSC}} \}
\] (1.15)

s.t.

\[
\left( t^C_{mh} - p^h_{mh} - t^C_{mj} \right) x^M_{mj} x^M_{mh} y_{jh} \geq 0, \forall j, h, m
\] (1.16)

\[
r_{jh} y_{hj} = 0, \forall j, h
\] (1.17)

\[
y_{jj} = 0, \forall j
\] (1.18)

\[
\sum_{m=1}^{M} x^M_{mj} = 1, \forall j
\] (1.19)

\[
x^M_{mj} = 0, \forall j \notin A_m, \forall m
\] (1.20)

\[
y_{jh} \in \{0, 1\}, \forall j, h
\] (1.21)

\[
x^M_{mj} \in \{0, 1\}, \forall m, j
\] (1.22)

\[
t^C_{mj} \geq 0, \forall m, j
\] (1.23)

Equation (1.15) is the objectives of minimization of total processing time \(t_P\), Equation (1.16) imposes that for any resource (machine), if cannot be selected for one operation until the predecessor is completed and also idle time must be considered. Equation (1.17) ensures that the precedence constraints are not violated. Equation (1.18) Equation
and [1.20] ensure the feasible resource selection. Equations [1.21], [1.22] and [1.23] impose nonnegative condition.

With the development of computers, computer-aided process planning (CAPP) systems have been used in order to aid manufacturing people to design process plans, thus reducing time and cost and increasing productivity. CAPP can assist human planner and achieve the integration of computer-aided design (CAD) and computer-aided manufacturing (CAM).

The PP is a combinatorial problem. For \( J \) operations to be processed in \( M \) machines, \( L \) tools, and \( D \) tool access directions, the total number of possible solutions could be \( ((J!^M)!^L)!^D \). It is difficult to optimize the process plans.

Since 1965, when Nieble [35] described the first computer-aided process planning (CAPP) system, numerous research efforts have been reported in this PP problem.

Generally, CAPP approaches can be classified into two categories [36]:

- **variant**
- **generative**

Variant approach makes process plan basing on a previous similar process plan. Most of the earlier CAPP systems can be categorized under the variant approach [37].

The generative approach can be used to automatically generate an optimal process plan according to the part’s features and manufacturing requirements. Most of the generative systems are knowledge-based systems utilizing artificial intelligence techniques. They are oriented toward the needs of large companies, especially those producing products with large variety and small batch sizes.

Various approaches to CAPP have been proposed in the literature [37], [38]. Research studies on PP include GA-based approaches [39], [40], [41], [50], and [42] neural-network-based approaches [43] and [44], knowledge-based approaches [45], [46] and [47], and agent-based approached [48], [49], [50], [51], [52], and [53].

### 1.3.2 Process Planning and Scheduling Problem

The PP is the determination of operations (machines, tools, and tool access directions) and their sequences for manufacturing a part effectively and economically. The core activity of PP is to decide which manufacturing resources to be selected and in which sequence to be used according to objective of achieving the correct quality, the minimal processing time and/or production cost. Scheduling is the allocation of the resources in the shop over time to manufacture the various parts [54]. It is used to determine the most appropriate moment to execute each operation for the launched production orders, taking into account the due date of these orders, a minimum makespan, a balanced resource utilization, etc., to obtain high productivity in the workshop [1], [2].

In MS, both PP and scheduling are responsible for the effective allocation and utilization of resources. For PP problem, it usually assumes that all the applicable manufacturing resources are available for this part. The scheduling problem is to handle a group of parts with temporal constraints and competitive manufacturing resources, and decide how and when to assign the manufacturing resources to the parts. In a complex manufacturing situation, it is ideal to integrate the PPS more closely to achieve a global optimum in manufacturing, and increase the production flexibility and responsiveness of the systems. It is the merit of integrated PPS.
The PPS is to process a set of parts with operations on machines by selecting suitable manufacturing resources (machines, tools and TADs) and sequence the operations so as to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved.

Given a set of parts and a set of manufacturing resources, the PPS problem can be defined as follows.

- **Operation sequencing**
- **Process selection**
- **Part scheduling**—Determine how and when to assign the manufacturing resources to the parts.

### 1.3.3 Cutting-Edge of the Research Survey

In the past decade, a number of research works have appeared to address the integration of PPS.

To facilitate the optimization process, some optimization approaches based on modern heuristic algorithms and artificial intelligence (AI) technologies, such as the genetic algorithm (GA)\[42\] \[55\] \[56\], simulated annealing (SA) algorithm \[1\] \[54\], Tabu search algorithm \[65\], agent-based approach \[66\] \[67\] and particle swarm optimization (PSO) algorithm \[34\] \[68\], have been developed in the last decade and significant improvements have been achieved.

**Genetic algorithm**

Morad and Zalzala developed a GA-based integration scheme, in which process plans have been represented as chromosomes, and crossover and mutations operations have been used to explore the alternative process plans. The performance criterion to choose a satisfactory schedule from the process plans can be the minimum makespan, the setup cost or the tardiness \[42\].

Kim et al. developed a single optimization model to integrate the PPS. In the work, three rules, which are operation flexibility, sequencing flexibility and processing flexibility, have been employed to generate multiple process plans. From these multiple plans, a symbiotic GA has been used to search for an optimized process plan that satisfies scheduling objectives, such as the minimum makespan or the mean flow time \[55\].

Moon and Seo proposed an advanced PPS model for the multi-plant. The objective of the model is to decide the schedules for minimizing makespan and operation sequences with machine selections considering precedence constraints, flexible sequences, and alternative machines. Numerous experiments are carried out to demonstrate the efficiency of the proposed approach \[56\].

Zhang and Yan developed an optimization model to combine the considerations from PPS, such as the production cost, the tardiness time, the set-up cost and the early finish time. Based on this, an improved hybrid GA-based approach has been designed to optimize planning and scheduling simultaneously \[57\].

Jia et al. proposed a modified genetic algorithm to solve multi-factory distributed scheduling problems. They suggested an encoding of chromosome, crossover mechanism, and two mutation mechanisms. Their modified GA has been compared with other classical scheduling and heuristic approaches, and obtained satisfactory results.
However, the size of the problems they used were small, and the algorithm can only handle fixed production routine [58][59].

Jia et al. presented a new approach to determine good combinations of factories (process plans) to manufacture the jobs and in the meantime generate good operation schedules. A genetic algorithm, integrated with Gantt chart (GC), is proposed to derive the factory combination and schedule. The integration of GA-GC is proved to be efficient in solving small-sized or medium-sized scheduling problems for a DMS. Multiple objectives can be achieved, including minimizing makespan, job tardiness, or manufacturing cost [60].

Chan et al. proposed a new idea, namely genetic algorithms with dominant genes (GADG) in order to deal with FMS scheduling problems with alternative production routing. The proposed GADG will identify and record the best genes and structure of each chromosome. A new crossover mechanism is designed to ensure the best genes and structures to undergo crossover. The performance of the proposed GADG is testified by comparing it with other existing methodologies, and the results show that it outperforms other approaches [61][63].

Chan et al. proposed a genetic algorithm with dominant genes to deal with distributed scheduling problems, especially in a flexible manufacturing system (FMS) environment [62].

According to prescribed criteria such as minimizing processing time and cost, a genetic algorithm is proposed by Li et al. applied in distributed environment. It is shown from the case study that the approach can obtain optimal or near-optimal process plan [64].

A unique method for the integration of PPS in a batch-manufacturing environment is reported by Zhang et al. [54]. This integration is essential for the optimum use of production resources and for the generation of realistic process plans that can be readily executed with little modification. The integrated system was implemented in the manufacturing of prismatic parts. The testing results show that the developed integration method can achieve satisfactory process plans and a schedule in an effective and efficient manner.

**Simulated annealing**

A unified representation model and a simulated annealing-based approach have been developed by Li and McMahon to facilitate the integration and optimization process. In the approach, three strategies, including processing flexibility, operation sequencing flexibility and scheduling flexibility, have been used for exploring the search space to support the optimization process effectively. Performance criteria, such as makespan, the balanced level of machine utilization, job tardiness and manufacturing cost, have been systematically defined to make the algorithm adaptive to meet various practical requirements. Case studies under various working conditions and the comparisons of this approach with two modern evolutionary approaches are given. The merits and characteristics of the approach are thereby highlighted [1].

**Tabu search**

Yan et al. developed an optimization model to combine the considerations from PPS, such as the production cost, the tardiness time, the set-up cost and the early finish
time. Based on this, a tabu search (TS)-based approach has been designed to optimize planning and scheduling simultaneously [65].

Agent-based approach

Wong et al. presented the development of an agent-based negotiation approach to integrate PPS (IPPS) in a job shop kind of flexible manufacturing environment [66]. The agent-based system comprises two types of agents, part agents and machine agents, to represent parts and machines respectively. An agent-based framework is established to simulate the proposed IPPS approach. The performance measures, including makespan and flow time, are compared with those of a search technique based on a co-evolutionary algorithm [67].

Particle swarm optimization

Guo et al. employed a particle swarm optimization (PSO) algorithm to optimize the integrated PPS problem. PSO algorithm has been enhanced with new operators to improve its performance and different criteria, such as makespan, total job tardiness and balanced level of machine utilization, have been used to evaluate the job performance. To improve the flexibility and agility, a re-planning method has been developed to address the conditions of machine breakdown and new order arrival. Case studies show that the developed PSO can generate satisfactory results in optimizing the IPPS problem [68].

1.3.4 Insufficiencies of Previous Researches

As to the other most important scheduling problem in machining, PPS looks highly complex due to its characteristics. Process planning (PP) is the determination of optimal process plans, i.e. operations (machine, tool, tool access direction (TAD)) and their sequences. All the manufacturing resources are assumed as available in this phase. The scheduling is determination of the most appropriate moment to execute each operation in the shop over time with competitive resources. Traditionally, PP and scheduling are performed sequentially as separate activities, where scheduling is carried out after generating the process plans accordingly. The conventional PPS may lead to the local optimum and low productivity in machining. Integrated PPS can present significant efficient improvements through eliminating scheduling conflicts, reducing flow-time, and improving resource utilization. Additionally, previous researchers mostly focused on single objective optimization rather than multiobjective optimization. Moreover, the existing moEAs cannot get satisfactory efficacy (convergence and dispersion) and efficiency (computation time) performances.

Furthermore, in recent years, many companies have changed their MS from traditional single-factory to multi-factory distributed manufacturing system (DMS) to increase their international competitiveness. Different factories located in different areas have different machining capacities and different resource constraints. Except of the allocation of resource to process the products which are assigned into only one factory, the assignment of products into factories evenly will influence the whole PPS. How to allocate the products into factories, how to arrange the resources for each factory to improve the productivity and balance the workload of factories become exceedingly difficult.
For PPS problem

- Previous researchers usually optimized PPS separately without consideration as integrality.
- Most of the above previous researchers just concern with satisfying single objective.
- Previous moEA approaches cannot get a good balance between efficacy and efficiency performances

For PPS problem in DMS

- There is a lack of papers, which have studied PPS problems in DMS.
- Previous researchers usually optimized PPS separately without consideration as integrality.
- Most of the above previous researchers just concern with satisfying single objective.
- Previous moEA approaches cannot get a good balance between efficacy and efficiency performances.

1.4 Research Objectives

Multiobjective single-model ALB (smALB) problem deals with the allocation of the proper tasks of one model among proper stations and assignment of the proper workers among proper stations in order to generate a set of Pareto optimal solutions satisfying multiple objectives simultaneously. The previous research works lack consideration of allocating tasks and workers among stations considering different workers’ capabilities, which might cause more idle time, reduction of the line efficiency, and high worker cost. Moreover, the previous approaches based on EAs need complex coding methods or special processes to avoid infeasible solutions. It causes the low efficiency. Therefore, issue 1 is optimization of multiobjective smALB problem efficiently with minimization objectives for cycle time, variation of workload of station, and worker cost concurrently considering various workers’ capabilities.

With the increasing and varied demands of customers at recent market, manufacturers are forced to produce a set of similar models of the same product on an assembly line so that the potential market will not be weakened. It has led to the multiobjective mixed-model ALB (mALB) problem. Therefore, issue 1 is extended to more complex issue 2, which is optimization of multiobjective mALB problem effectively with same objectives as issue 1 considering workers’ capabilities. Different from issue 1, calculation of cycle time with multiple models becomes more difficult than single model.

Multiobjective PPS in MS processes a set of parts with operations on machines by selecting suitable manufacturing resources, sequencing the operations, and determining the most appropriate moment to execute each operation to generate a set of Pareto optimal solutions satisfying multiple objectives simultaneously. Among scheduling problem, PPS looks extremely complicated due to its characteristics. Process planning is the determination of optimal process plans, i.e. operations (machine, tool, tool
access direction) and their sequences. All the manufacturing resources are assumed as available in this phase. The scheduling is determination of the most appropriate moment to execute each operation in the shop over time with competitive resources. Traditionally, process planning and scheduling are performed sequentially as separate activities, where scheduling is carried out after generating the process plans accordingly. The conventional PPS may lead to the local optimum, low productivity in machining. Integrated PPS can present significant improvements to the efficiency of machining through eliminating scheduling conflicts, reducing flow-time, and improving resources utilization. Previous researchers usually optimized PPS separately without consideration as integrality. They mostly focused on single objective rather than multiobjective optimization. The previous research works use EAs to solve PPS; however, the proposed approaches cannot get a good balance between efficacy (quality, i.e., convergence and dispersion) and efficiency (speed, i.e., computation time) performances. Therefore, issue 3 is efficient optimization of multiobjective PPS with minimization objectives for makespan, variation of workload of machine concurrently.

Within recent years, manufacturing is gradually changing to a distributed manufacturing system (DMS), which have different capabilities and constraints. In recent years, many companies have changed their MS from traditional single-factory to multi-factory distributed manufacturing system (DMS) to increase their international competitiveness. Different factories located in different areas have different machining capacities, different resource constraints. Each machine and tool in each factory have respective machining time and cost. Except of optimization of PPS in one factory, allocation of products into factories should be concerned carefully in order to optimize entire PPS as parallel in DMS. How to allocate the products into factories, how to arrange the resources for each factory to process products, how to achieve a number of global optimums in machining to improve the productivity, balance the workload of factories becomes more difficult. It leads issue 3 extending to issue 4, effective optimization of multiobjective PPS in DMS with minimization objectives for maximum total processing time, maximum variation of workload of machine at the same time.

1.5 Methodology and Limitations

1.5.1 Methodologies

Because the ALB problem and PPS problem in MS belong to classical combinatorial optimization problem. These two problems have been proved which fall into the NP-hard class of problem. Moreover, multiobjective optimization makes these two problems more difficult and complex. The traditional optimum seeking methods have been proved not to be of practical use for large problems due to their computational inefficiency.

In recent years, to provide an alternative to traditional optimization techniques, numerous research efforts have been directed towards the development of evolutionary algorithms (EA), which is a subset of evolutionary computation (EC), a generic population-based meta-heuristic optimization algorithm in artificial intelligence (AI). Evolutionary algorithms use some mechanisms simulated the biological evolution: reproduction, mutation, recombination, and selection. Candidate solutions stand for the role of individuals of populations, and fitness function determines the environment within
which the solutions survive. In selection, individuals for the next generation are chosen by a bias towards higher fitness. The individuals reproduce by recombination and/or mutation. Recombination acts as the two selected candidates (parents) and results in one or two new candidates. Mutation acts as one candidate (parent) and results in a new candidate. These operators create the offspring (a set of new candidates). These new candidates compete with old candidates for their place in the next generation (survival of the fittest). This process can be repeated until a candidate with sufficient quality (a solution) is found or a predefined computational limit is reached.

Evolutionary algorithms include genetic algorithm (GA), genetic programming (GP), evolutionary programming (EP), evolution strategy (ES), differential evolution (DE), particle swarm optimization (PSO), ant colony optimization (ACO), invasive weed optimization algorithm, harmony search and Gaussian adaption.

Evolutionary algorithms seem to be especially suited to multiobjective optimization because they are able to explore multiple Pareto-optimal solutions in a single simulation run and may exploit space of solutions by recombination. Therefore, this thesis focuses on investigating moEAs to solve the multiobjective scheduling problems in machining and assembly processes.

1.5.2 Limitations

This dissertation is based on academic research. The data and information used was obtained from electronic sources, annual reports, books and other published and unpublished journals. Some problem-dependent information is added to describe the characteristics of problem. For real problems in MS, more constraints need to be considered and all of approaches may be need to be reconstructed to adapt the real demands. No one approach could solve all the problems.

Comparative analysis on performance of different algorithms is based on some performance indices, not all. It is not necessary to compare the results with all of performance indices. Moreover, the selection of the performance indices is based on the importance and the authors’ interests. This selection can affect the proper comparison of different algorithms. However, data limitations do not, in any way, invalidate the findings of the research.

1.6 Outline of the Dissertation

In Chapter 1, background and motivation of this research are described firstly. Following, the definition, characteristics, methodology, classification and cutting-edge of research survey of ALB problem and PPS problem are introduced. Based on related research works, the objectives of this research are proposed. Last, methodologies for solving this problem are introduced in this Chapter.

In Chapter 2, some basic concepts and terminologies of multiobjective evolutionary algorithms (moEA) are introduced firstly. Next, Some classical multiobjective evolutionary algorithms are introduced to provide some technical support or reference for comparison in later research. Some common performance indices to evaluate the performance of different multiobjective evolutionary algorithms are listed in the last of this Chapter.

Chapter 3 presents a random key-based genetic algorithm (rkGA) to deal with the smALB problem with considering workers’ capabilities. Three objectives, minimiza-
tion of cycle time, minimization of station workload and minimization of total worker cost, are taken into account. New mathematical models for describing the three objectives of smALB are introduced firstly. After, the genetic representation, crossover and mutation operators, selection and fitness function of the proposed methods are described in detail. The performance of proposed method is validated through numerical experiments for solving the 35 tasks, 6-15 stations and 6-15 workers ALB problem. The experimental analysis indicate the efficacy and efficiency performances of the proposed approach.

Chapter 4 describes a generalized Pareto-based scale-independent fitness function (gp-siff) based genetic algorithm (gp-siffGA) to solve the mALB problem with considering workers’ capabilities. Same three objectives, minimization of cycle time, minimization of station workload and minimization of total worker cost, are considered in this Chapter. New mathematical models for describing these three objectives of mALB are suggested firstly. The composition and decomposition of models is proposed to simply the problem. Accordingly, a novel reference cycle time calculation method according to different demand ratio of models is proposed. As fitness function, gp-siff is used to enhance the convergence performance and reduce the computation time. The genetic representation, crossover and mutation operators, selection of the proposed methods are described in detail. The performance of proposed method is validated through numerical experiments for solving the mALB problems with 35 tasks (3 models, 4, 5, 7, and 12 stations), 70 tasks (5 models, 7, 10, 14, and 19 stations), 148 tasks (7 models, 10, 14, 21, and 29 stations). The experimental analysis demonstrate the performance of the proposed approach.

Chapter 5 presents a fast and effective multiobjective evolutionary algorithm (FE-MOEA) to deal with PPS problem. Two minimization objectives for makespan, variation of machine workload are considered concurrently. The genetic representation consists of operation sequence, machine, tool and tool access direction vectors. A new Pareto dominating and dominated relationship-based (PDDR) fitness function is proposed to evaluate individuals. PDDR has a clear advantage than most famous moEAs, NSGA-II and SPEA2 on time complexity. PDDR has also an advantage with the tendency converging toward the central area of the Pareto front. The first moEA, Vector evaluated genetic algorithm (VEGA), has an advantage of VEGA with the edge region of the Pareto front due to its selection only for one objective. These two advantages are united tactfully in FE-MOEA. These two mechanisms not only preserve the convergence rate, but also guarantee the better dispersion performance. The order-based crossover, resource-based mutation, local search and BTS are adopted to improve the performance. The proposed method is validated by 4 parts with 57 operations PPS problem. The experimental analysis indicate that the efficacy and efficiency performances of the proposed approach have been improved clearly.

Chapter 6 introduces a fast multiobjective genetic algorithm with archive (fmoGA-A) mechanism to deal with multiobjective PPS problem in DMS. Two minimization objectives for maximum total processing time, maximum variation of machine workload are concerned. The multiple vectors genetic representation is proposed to optimize the process planning and scheduling simultaneously. For maintaining a set of nondominated individuals found up-to-now, the elitism mechanism named archive is proposed. A gp-siff with less time complexity is adopted to evaluate the fitness function. According to the gp-siff values, the archive has been updated in evolving process. The gp-siff-based archive mechanism is used to improve the quality and
speed. After, crossover and mutation operators, selection of the proposed methods are described in detail. The proposed method is validated by 2 factories, 4 parts PPS problem. The experimental analysis indicate the efficiency performance of the proposed approach has been increased significantly and the efficacy performance is not less than NSGA-II and SPEA2.

Finally, Chapter 7 provides a conclusion of this study and guides a discussion to future research.
Chapter 2

Multiobjective Evolutionary Algorithms

2.1 Multiobjective Problems

2.1.1 Basic Concepts

In single objective optimization problems, it is easy to compare objective value according to categories: one is better than the other is, vice versa. However, almost every real world problem involves optimization of several incomparable or incommensurable and often competing objectives simultaneous. For multiple objectives problems, there exist many "good" points, which are not worse than any point in the objective space. Single objective can be as a special multiple objectives, i.e. the preference is one objective no matter how much other objectives are. Therefore, the single objective is compromised, the multiple objectives is actual.

For example, consider to design a complex hardware/software system such as note PC, mobile phones, cars, etc. Often the cost of such systems is to be minimized, while maximum performance is desired.

Multiple objectives problems arise in the design, modeling, and planning of many complex real systems in the areas of industrial production, urban transportation, capital budgeting, forest management, reservoir management, layout and landscaping of new cities, energy distribution, etc.

2.1.2 Terminologies

A multiple objectives problem is usually given in the following form:

\[
\begin{align*}
\text{minimize} \quad & \{z_1 = f_1(x), z_2 = f_2(x), \ldots, z_m = f_m(x)\} \\
\text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, q \\
& h_j(x) = 0, \quad j = q + 1, \ldots, k \\
& x \in \mathbb{R}^n
\end{align*}
\] (2.1)

where \(x\) is the decision variable, \(f_i\) is objective \(i\), \(g_i\) is inequality constraint \(i\), and \(h_j\) is equality constraint \(j\).

There are \(m\) objective need to be optimized in Eq. 2.1 which makes the model a multiobjective optimization problem (moOP).

Two spaces need to be defined moOP.
Parameter space

Parameter space (or decision space) defines as follows:

\[ S = \{ x \in \mathbb{R}^n \mid g_i(x) \leq 0, \ i = 1, \cdots, q \cup h_j(x) = 0, \ j = q + 1, \cdots, k \} \] (2.2)

where \( x \in S \) is called a parameter vector (or decision vector).

Objective space

Objective space (or criterion space, fitness space) is defined by Eq. 2.3. \( z \in Z \) is called an objective vector.

\[ Z = \{ z \in \mathbb{R}^m \mid z_1 = f_1(x), z_2 = f_2(x), \cdots, z_m = f_m(x) \} \] (2.3)

Dominance

The most important terminology given by Pareto is “dominance”, which can be defined in either objective space or parameter space.

In objective space, point \( i \) dominates point \( j \) means

\[ z_i^l \leq z_j^l, \ \forall l \in \{1, 2, \cdots, m\} \]
\[ z_i^l < z_j^l, \ \exists k \in \{1, 2, \cdots, m\} \] (2.4)

where \( z_i^l \) is the \( l \)’th objective value of point \( i \).

In parameter space, point \( i \) dominates point \( j \) means

\[ f(x_i)^l \leq f(x_j)^l, \ \forall l \in \{1, 2, \cdots, m\} \]
\[ f(x_i)^k < f(x_j)^k, \ \exists k \in \{1, 2, \cdots, m\} \] (2.5)

where \( f(x_i)^l \) is the objective \( l \)’s value of the point \( i \).

\( i \prec j \) is used to demonstrate that point \( i \) dominates \( j \) (only means that \( i \) is “better” than \( j \)).

- If \( z_i^l = z_j^l, \ \forall l \in \{1, 2, \cdots, m\} \), means \( i \) is equivalent to \( j \), expressed as \( i = j \).
- If \( i \) either dominates or is equivalent to \( j \), means \( i \) covers \( j \).
- If \( z_i^l \leq z_j^l, \ \forall l \in \{1, 2, \cdots, m\} \), means \( i \) weakly dominates \( j \), expressed as \( i \preceq j \).
- If \( z_i^l < z_j^l, \ \forall l \in \{1, 2, \cdots, m\} \), means \( i \) strongly dominates \( j \), expressed as \( i \ll j \).

Nondominated solution

In the objective space, those points, which are not dominated by any other point are most cared. Therefore, the definition of nondominated solution (or Pareto optimal solution) is addressed as follow.

A point \( z^0 \in Z \) in objective space is a nondominated solution if and only if there isn’t any point \( z \in Z \) which dominates \( z^0 \).
Pareto front

All the nondominated points consist of the Pareto front\(^1\), often expressed as PF\(^*\).

Efficient solution

A point \(x_0 \in S\) in parameter space is an efficient solution (or noninferior solution) if and only if there isn’t any point \(x \in S\) which dominates \(x_0\). All the effective solutions consist of the Pareto set\(^2\), often expressed as P*. PF* is the map of P*.

2.2 Multiobjective Evolutionary Algorithms

Apart from single objective optimization problem with a single optimal solution, moOPs generally have many solutions with incommensurable quality for designers and decision makers. Due to some features of moOP hard to be quantified, i.e. mathematically modeled, designers and decision makers like the multiple solutions with same good quality so that they could select from them according to their objective preferences. On the other hand, special concerns are necessary to handle the solutions, especially when the number of the nondominated solutions is of substantial size. So an algorithm is considered to be efficient if it could provide adequate number of nondominated solutions in a single run.

Evolutionary algorithms (EAs) (EAs are also called evolutionary computation (EC) and evolutionary intelligence (EI).) are a class of stochastic optimization methods that simulate the process of natural evolution.

The main characteristics of EAs are: (1) population-based, EAs maintain a group of solutions, denoted as population, to optimize or learn the problem in a parallel way. Population is the basics of the evolving process. (2) fitness-oriented, Every solution in the population is called individual. Individual has its gene representation, denoted as code, and performance evaluation, denoted as fitness value. EAs prefer fitter individuals, which is the basics of the optimizing and converging of the algorithms. (3) variation-driven, Individuals will undergo various variation operators to mimic the genetic gene changes, which is the basics of searching the solution space.

From above mentioned, the most distinguishing feature of EAs is that a group of individuals evolve together to search the solution landscape in a parallel way. So if EAs could control the diversity of the population by avoiding all individuals to converge to one point and encouraging them to reside in different points on Pareto front, They could provide the designers with at most popsize nondominated solutions in a single run. Therefore, EAs seem to be especially suited to solve moOP.

Evolutionary algorithms designed for moOPs are called multiobjective evolutionary algorithms (moEAs), and this kind of optimization is called evolutionary multiobjective optimization (EMO).

2.2.1 Classical Multiobjective Evolutionary Algorithms

After the first pioneering studies on moEAs appeared in the mid-1980s, they have been receiving growing interest from researchers with various backgrounds. Some

\(^1\)Other names include Pareto frontier and Pareto optimal front.

\(^2\)Other names include Pareto solution set and Pareto optimal set.
classical moEAs have been proposed and proved to resolve moOP efficiently, such as adaptive weight genetic algorithm (awGA), Vector Evaluated Genetic Algorithm (VEGA), Nondominated Sorting Genetic Algorithm II (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA2).

Adaptive Weight Genetic Algorithm

Approaches to solve multiple objectives could be divided roughly into two parts: preference-based approaches and generating approaches. If there are some preference information, which objective decision maker cares more, before optimization, preference-based approaches could embed the user requirement into the algorithms and generate high qualified solutions quickly. However, under the blind circumstances, generating approaches could provide multiple nondominated solutions for users to make decisions.

Generally, preference-based approaches transfer the moOPs into single objective problems in different ways. Traditionally, a weight related to the importance for each objective is given. Through accumulating the objectives with weight, multiple objectives can be transfer into one objective simply.

However, in most situations, for example, there are two objectives. the value of objective 1 is in the interval of \([10^2 \sim 10^7]\). The value of objective 2 is in the interval of \([0.01 \sim 0.77]\). If simple sum method is used to do the optimization, the value of objective will be ignored in sum value. The main focus is on objective 1 and there is little concern for objective 2, which makes the exploration process for objective 2 incomplete.

It is hard to define the objective function value interval before the optimization process. If the range for all objective values are known, they can be normalized to the range \([0, 1]\) and appoint weights on objectives for fixed weight sum method. However, EAs are population-based algorithms which contains a group of individuals. The population of the current generation represents some kind of sampling over the solution space. Therefore, the approximate objective function value interval can be gotten according to current population.

Gen and Cheng [27] proposed an adaptive weights approach which utilizes some useful information from current population to readjust weights in order to obtain a search pressure towards to positive ideal point (for maximum problem). For the examined solutions at each generation, we define two extreme points: the maximum extreme point \(z^+\) and the minimum extreme point \(z^-\) in the criteria space as the follows:

\[
\begin{align*}
  z^+ & = \{z_{1\text{max}}, z_{2\text{max}}, \cdots, z_{q\text{max}}\} \\
  z^- & = \{z_{1\text{min}}, z_{2\text{min}}, \cdots, z_{q\text{min}}\}
\end{align*}
\]

where \(z_{k\text{max}}\) and \(z_{k\text{min}}\) are the maximal value and minimal value for objective \(k\) in the current population. Let \(P\) denote the set of current population. For a given individual \(x\), the maximal value and minimal value for each objective are defined as the follows:

\[
\begin{align*}
  z_{k\text{max}} &= \max \{f_k(x) | x \in P\}, \quad k = 1, 2, \cdots, q \\
  z_{k\text{min}} &= \min \{f_k(x) | x \in P\}, \quad k = 1, 2, \cdots, q
\end{align*}
\]

The hyper parallelogram defined by the two extreme points is a minimal hyper parallelogram containing all current solutions. The two extreme points are renewed at
each generation. The maximum extreme point will gradually approximate to the positive ideal point. The adaptive weight for objective $k$ is calculated by the following equation:

$$w_k = \frac{1}{z_{k}^{\max} - z_{k}^{\min}}, \quad k = 1, 2, \cdots, q$$

(2.10)

For a given individual $x$, the weighted-sum objective function is given by the following equation:

$$z(x) = \sum_{k=1}^{q} w_k(f_k(x) - z_{k}^{\min})$$

$$= \sum_{k=1}^{q} \frac{f_k(x) - z_{k}^{\min}}{z_{k}^{\max} - z_{k}^{\min}}$$

(2.11)

As the extreme points are renewed at each generation, the weights are renewed accordingly.

If there really exists a positive ideal point, its normalized objective is $q$. Similarly, the normalized objective for negative ideal point is 0. In this way, every individual could be mapped into the interval $[0, q]$. The bigger $z(x)$ is, the better $z(x)$ is.

Vector Evaluated Genetic Algorithm

The original idea of applying EAs to moOP came from Schaffer’s vector evaluated genetic algorithm (VEGA) suggested in 1985 [69].

For example, two objectives problem, VEGA is straightforward: dividing the population into 2 sub-populations, each of which evolve toward only a single objective. The whole algorithm could be illustrated by Fig. 2.1.

![Figure 2.1: Evolving process in one generation of VEGA](image)

VEGA evolves the objectives in a parallel way. The selection process in Fig. 2.1 could be any one selection method because there is only one objective need to be considered. The shuffle procedure is quite necessary when the crossover and the mutation is carried out from individual 1 to $\text{popsize}$. In this way, individuals from different sub-populations have chances to exchange information, which promote the exploration of the objective space.

The advantage of VEGA is less time complexity because it doesn’t need transform multiobjectives values into one value, doesn’t need calculate the dominate level of
each individuals basing on dominated relationship. However, the characteristics of VEGA cause selection bias. The quality of obtained solutions will not be good because of the drawback of VEGA. VEGA is not good at directly distributing individuals along PF*.

Nondominated Sorting Genetic Algorithm II

In 1994, Srinivas and Deb proposed their nondominated sorting genetic algorithm (NSGA) by combining Goldberg’s Pareto rank method and fitness sharing together. After Pareto ranking, every individual with the same rank \( r \) gets the same dummy fitness value \([70]\). Then individuals share their fitness value with others in the same rank. In this way, lower rank (good quality in Pareto dominance) individuals win the selection, which promote NSGA toward the PF* with good distribution.

Deb et al. suggested NSGA-II in 2002 to improve NSGA in three ways: improving the Pareto ranking procedure to make it with less time complexity; adding the elitism mechanism; and pruning the individuals by the new crowding distance method \([71]\).

In NSGA-II, the capacity of the archive is \( \text{popsize} \). The solution process of one generation in NSGA-II is as follows.

**One generation of NSGA-II**

**Phase 1**: Calculating Pareto rank and crowding distance.
- Step 1.1: Combine the population \( P(t) \) (\( \text{popsize} \)) and the archive \( A(t) \) (\( \text{popsize} \)) to get \( 2 \times \text{popsize} \) individuals.
- Step 1.2: Calculate the rank of each individual.
- Step 1.3: Calculate the crowding distance for each individual.

**Phase 2**: Generating the new archive \( A(t+1) \).
- Step 2.1: Insert the individuals into \( A(t+1) \). The individuals in rank 1 should be inserted first, then rank 2 ... If rank \( r \) cannot be fully inserted into \( A(t+1) \), insert the individual in the descending order of the crowding distance until \( A(t+1) \) is full (With \( \text{popsize} \) individual).

**Phase 3**: Generating the new population \( P(t+1) \).
- Step 3.1: Select from \( A(t+1) \) using binary tournament selection to form mating pool. If two individuals in \( A(t+1) \) have different ranks, the one with lower rank wins the tournament. Alternatively, the one with same rank but larger crowding distance wins the tournament.
- Step 3.2: Generate the new population \( P(t+1) \) by crossover and mutation from mating pool.

Fig. 2.2 illustrates the evolving process in one generation of NSGA-II.

If an individual \( j \) is residing in PF* and far from others, it will never be lost. It will always be in rank 1 with large crowding distance which means it will absolutely in \( A(t+1) \).

The convergence mechanism of NSGA-II is Pareto rank and tournament selection; the distribution mechanism of which is crowding distance; and the elitism mechanism of which is the archive \( A \).
Strength Pareto Evolutionary Algorithm 2

Zitzler et al. suggested strength Pareto evolution algorithm (SPEA) in 1999 and improved it into SPEA2 in 2001 [72].

The evolving process in one generation of SPEA2 could be illustrated by Fig. 2.3. $A(t)$ represents the archive at generation $t$ with size $\bar{N}$ and $P(t)$ represents the population at generation $t$ with size $N$.

In SPEA2, every individual has been assigned a number to describe its “strength” as follows

$$S(i) = |\{j \in P + A \cap i < j\}|$$  \hspace{1cm} \text{(2.12)}

where $| |$ is the function of cardinality, i.e. the number of elements in one set. According to Eq. 2.12 an individual’s strength is the number of individuals it dominates in the union of $A$ and $P$. The larger the number is, the stronger the individual is.
Only comparing the strength of every individual will lead to selection bias. Zitzler et al. defined another term **raw fitness** to describe how good an individual is in the performance of convergence. An individual’s raw fitness is defined as follows:

\[ R(i) = \sum_{j \in P + \mathbb{A}, j \prec i} S(j) \]  

(2.13)

The smaller raw fitness is, the better convergence performance will be.

On dispersion, in SPEA2, the distances of every individual to all other individuals are necessary. Then the distances of one individual are sorted in ascending. Zitzler et al. defined a term **density** to describe the crowdness of individual \( i \) as follows.

\[ D(i) = \frac{1}{s_{ik}^k + 2} \]  

(2.14)

where \( s_{ik}^k \) is the \( k \)’th shortest distance to individual \( i \), i.e. the \( k \)'th number in the distance rank. The further an individual is away from others, the larger \( s_{ik}^k \) is, and the smaller \( D(i) \) is. Zitzler et al. suggested that \( k = \sqrt{N + N} \) might be a good parameter set.

“Density” here describe the crowdness of an individual. If evenly distributed individuals are preferred, the density of the individual should always be small.

Finally, every individual is granted the fitness by Eq. 2.15, which constitutes the bases of the binary tournament selection in Fig. 2.3

\[ F(i) = R(i) + D(i) \]  

(2.15)

For nondominated individual \( i \), \( R(i) = 0 \) and \( D(i) < 1 \). Therefore, the moOP has been transferred into a single objective minimum problem in this way.

Let \( B(t) \) represents the nondominated set selected from the union of \( A(t) \) and \( P(t) \) and \( |B(t)| \) represents its cardinality.

If \( |B| < \bar{N} \), more dominated individuals are added to fill in the gap according to fitness described by Eq. 2.15 until \( |B| = \bar{N} \).

If \( |B| > \bar{N} \), individuals from \( B(t) \) with the smallest distance are deleted to others \( (s_{ik}^1) \), then delete individuals with the smallest second distance to others \( (s_{ik}^2) \) …until \( |B| = \bar{N} \).

For dominated individuals, their raw fitness values are several, dozens, or hundreds, large enough to neglect their density, which is less than 0.5. So the SPEA2 direct the search toward the PF*. For nondominated individuals, their raw fitness values are all 0, so the sparser individual, the one with smaller density, gain the advantage.

Suppose individual \( j \) is residing in PF* and far from others. From the algorithms and Fig. 2.3 above, individual \( j \) will never be lost. Therefore, the elitism mechanism in SPEA2 works fine.

The convergence mechanism in SPEA2 is raw fitness assignment; the dispersion mechanism of which is density; and the elitism mechanism of which is also the introduction of the archive \( \mathbb{A} \).

Recently nearly, every new published paper on moEAs algorithm design and improvement needs to compare the suggested one at least with NSGA-II and SPEA2 on benchmark problems, which makes these two algorithms actually the benchmark algorithms in moEAs.
2.2.2 Performance Indices

How to evaluate the performance of different moEAs is very important decision in moOP. Those criteria are performance indices (PI).

The moEA need to be converged to PF*, therefore the convergence is often the first consideration need to be cared. How many individuals belong to PF* and what is the overall evaluation of convergence to PF* are two important things in convergence performance.

Distribution evenly along PF* is also one of the reasons using moEAs in solving moOP. Apart from that, spread performance evaluates the moEAs’ ability of capturing the edge points of PF*, which is sometimes being taking into consideration.

There might be several situations while evaluating an algorithm. First, PF* of the benchmark moOPs is known and want to evaluate how good the algorithm is while optimizing the benchmark moOPs. Second, PF* is not known a priori, but compare to the solutions of many algorithms is need. Therefore, the nondominated solutions in the union of these final solutions constitute a reference set, denoted as RS. Results from different algorithms are evaluated with RS. The PI for these two situations are called a unary index (input of the PI is one solution set). For an algorithm, all the nondominated solutions in the archive compose $S$.

The third situation is that just like to compare with two results ($S_1$ and $S_2$), or compare with many different results pair-by-pair, then need the PI to point out which one is better, sometimes how much better. The PI for the third situations is called a binary index (inputs of the PI are two solution sets).

In the comparison of two algorithms, the number of the objective evaluation for each algorithm should be the same to ensure the same search endeavor over the objective space and their archive should have the same capacity to ensure the same ability of reporting the nondominated solutions. For algorithm 1, all the nondominated solutions in the archive compose $S_1$. So does $S_2$.

Also need to be pointed out again is that the selection of the PI to be introduced is based on the importance and the authors’ interests.

Cardinality-based PI

If already PF* is known, or RS, the quality could be evaluated by counting how many individuals in $S$ are in PF*.

**Number of obtained solutions**

Evaluate each solution set depend on the number of obtained solutions (as shown in Fig. 2.4).

$$\text{NOS (S)} = |S|$$  \hspace{1cm} (2.16)

**Error ratio**

Van Veldhuizen suggested a unary PI named as error ratio (ER) in 1999 [73].

$$\text{ER (S)} = \frac{\sum_{i=1}^{|S|} e_i}{|S|}$$  \hspace{1cm} (2.17)

where $e_i = \begin{cases} 0 & z_i \in \text{PF}* \\ 1 & z_i \notin \text{PF}^* \end{cases}$ and $|S|$ is the cardinality of $S$. $\text{ER}$ could be calculated based
Multiobjective Evolutionary Algorithms

![Figure 2.4: Number of obtained solutions](image)

Figure 2.4: Number of obtained solutions

on RS too (as shown in Fig. 2.5). ER is to evaluate how bad a result is. Therefore, the smaller ER is, the better.

![Figure 2.5: Error ratio](image)

Figure 2.5: Error ratio

**Ratio of nondominated solutions**

This measure simply counts the number of solutions, which are members of the Pareto-optimal set RS (as shown in Fig. 2.6). The \( RNS(S) \) measure can be written as follows:

\[
RNS(S) = \frac{|S - \{x \in S | \exists r \in PF^* : r \prec x\}|}{|S|}
\]  

(2.18)

where \( r \prec x \) means that the solution \( x \) is dominated by the solution \( r \). An \( RNS(S)=1 \) means all solutions are members of the Pareto-optimal set \( PF^* \), and an \( RNS(S)=0 \) means no solution is a member of the \( PF^* \).

It is an important measure that although the number of obtained solutions \( |S| \) is large, if that the nondominated solutions ratio \( RNS(S) \) is 0, it may be a worst result. The difficulty with the above measures is that although a member of \( S \) is Pareto-optimal, if that solution does not exist in \( PF^* \), it may be not counted in \( RNS(S) \) as a non Pareto-optimal solution. Thus, it is essential that a large set for \( PF^* \) is necessary in the above equations. This PI is the opposite of the error ratio. Therefore, the bigger \( RNS \) is the better.

**Coverage**
Zitzler suggested a binary PI named as coverage (C) in 1999 [74].

\[
C(S_1, S_2) = \left| \frac{\{s_2 \in S_2 : \exists s_1 \in S_1 : s_1 \preceq s_2\}}{|S_2|} \right|
\]  

(2.19)

C (S_1, S_2) is the percent of the individuals in S_2 who are weakly dominated by S_1. The larger C (S_1, S_2) is, the better S_1 outperforms S_2 in C. It’s necessary to point out that generally C (S_1, S_2) + C (S_2, S_1) ≠ 1. C (S_1, S_2) and C (S_2, S_1) have been illustrated in Fig. 2.7.

**Volume-based PI**

Volume-based PI evaluate the performance of the solutions according to the size of covered by S.

**Hypervolume**

Zitzler suggested a unary PI named as hypervolume (HV) in 1999 [74]. The hypervolume is the size of the dominated space by S. In calculating HV, the reference point for computing need to be pointed out, this has been illustrated by o in Fig. 2.8. Then the size of the dominated spaces by S_1 could be the size enclosed by dotted line, and
that by $S_2$ is the size enclosed by dash line (as shown in Fig. 2.8). The larger $HV$ of a solution set is, the better it is. Knowles and Corne reported in 2003 that the selection of the reference point might influence the relative relationship between two sets of nondominated solutions, i.e. $HV(S_1)$ might be larger than $HV(S_2)$ with one reference point but smaller than $HV(S_2)$ with the other reference point [75].

![Figure 2.8: Hypervolume $HV(S_1)$ and $HV(S_2)$](image)

**Coverage difference**

To make the difference more obvious, Zitzler suggested another binary PI named as coverage difference ($D$) in 1999 [74].

$$
\begin{align*}
D(S_1, S_2) &= HV(S_1 + S_2) - HV(S_2) \\
D(S_2, S_1) &= HV(S_1 + S_2) - HV(S_1)
\end{align*}
$$

The larger $D(S_1, S_2)$ is, the better $S_1$ outperforms $S_2$ in $D$.

**Hypervolume ratio**

Van Veldhuizen suggested a similar PI named as hypervolume ratio ($HR$) in 1999 [73].

$$
HR(S_1, S_2) = \frac{HV(S_1)}{HV(S_2)}
$$

The larger $HR(S_1, S_2)$ is, the better $S_1$ outperforms $S_2$ in $HR$.

Because hypervolume-based PI is widely used and the calculation of $HV$ is complicated in high dimension situation, many efficient ways has been suggested recently [75–77].

**Distance-based PI**

Distance-based PI evaluate the performance of the solutions according to the distance to $PF^*$, or $RS$.

Van Veldhuizen suggested a unary PI named as generational distance (also named average distance) ($GD$) in 1999 [73]. First define the minimum distance from $S$ to $PF^*$ as

$$
d_i = \min_{p \in PF^*} \left\{ \sqrt{\sum_{k=1}^{m} (z_k^i - z_k^p)^2} \right\}
$$

(2.22)
where \( z_i^k \) is the \( k \)th objective value of the \( i \)th individual and \( d_i \) is its minimum Euclidean distance to \( PF^* \) (as shown in Fig. 2.9). Other norms, i.e. distance measure method, could also be used. \( GD \) is defined as follows:

\[
GD(S) = \left( \frac{1}{|S|} \sum_{i=1}^{|S|} d_i^q \right)^{1/q}
\]  

(2.23)

where \( q \) is a parameter. If \( q = 1 \), \( GD \) is equal to Deb’s \( \gamma \) PI [71] and Zitzler’s \( M_1^\gamma \) PI [74]. The smaller \( GD \) for one solution set is, the better it is in approaching \( PF^* \).

![Figure 2.9: Generational distance](image)

**Dispersion PI**

The one of dispersion PI is suggested by Schott in 1995 [78]. Schott named it as spacing (\( SP \)), which is the standard deviation of the closest distances.

\[
d_i = \min_{s_j \in S \cap s_j \neq s_i} \sum_{k=1}^{m} |z_i^k - z_j^k|
\]  

(2.24)

where \( d_i \) is the smallest distance from \( i \)th individual in \( S \) (as shown in Fig. 2.10). Then the average smallest distance of all individuals is calculated as follows:

\[
\bar{d} = \frac{1}{|S|} \sum_{i=1}^{|S|} d_i
\]  

(2.25)

\( SP \) is the standard deviation of \( d_i \).

\[
SP(S) = \sqrt{\frac{1}{|S|-1} \sum_{i=1}^{|S|} (d_i - \bar{d})^2}
\]  

(2.26)

Smaller \( SP \) might mean better dispersion performance.

For the above PIs, the following notices need to mentioned. [79]:

---

---
• No single PI is able to account for all aspects of the quality of moEAs.

• By unary indices, only means that algorithm 1 is no worse than algorithm 2 if the PI of algorithm 1 is better than that of algorithm 2.

The above mentioned PIs, only part of PIs proposed by researchers, are selected for this study. In this research, comparative analysis of different algorithms is based on some performance indices, not all. It is not necessary to compare the results with all of performance indices. Moreover, the selection of the performance indices is based on the importance and the authors’ interests. This selection can affect the proper comparison of different algorithms.
Chapter 3

Multiobjective smALB Considering Workers’ Capabilities

This Chapter is organized as follows: Section 3.1 introduces the background and related research works of problem. Section 3.2 describes the problem and assumptions. The mathematical formulations for this problem are extracted in Section 3.3. Section 3.4 introduces the details of approach to resolve this problem and experiments and discussion are addressed in Section 3.5. Last Section 3.6 summaries this Chapter.

3.1 Introduction

The ALB problem is a well-known manufacturing optimization problem, which determines the assignment of various tasks to an ordered sequence of stations, while optimizing one or more objectives without violating restrictions imposed on the line. The traditional ALB problem is to process one model with tasks by allocating the tasks into stations and assigning the worker into stations to generate an optimal solutions (schedule), which tells a production facility what to make, when, and on which equipment, which worker, to process the products effectively.

Seeking an optimal solution rapidly and effectively from all of the permutations, combinations of all of the tasks, manufacturing resources, human resources according to specified criteria is very difficult.

Moreover, in the modern industry, due to variety in design, there exist various products in a factory and the product mix changes greatly. On the other hand, workers in a factory are composed of regular members and contract-base employees, so that replacement of workers may occur once or twice a year. The fresh workers are assigned to stations after basic training, such as how to operate some machines or how to perform basic simply work tasks. After that, they have to improve their operational work experience during on-the-job training. Therefore, according to different work experiences, the processing time of a worker in a given task differs greatly among workers. It leads the ALB problem more complex to allocate tasks into stations and assign workers into stations.

Usually in a factory, there are a few skilled workers who can deal with the tasks on a high work experience. Therefore, the efficiency of the line can be improved by reducing the task times. Furthermore, operating the line causes short-term operating costs such as wages, material, set-up, inventory and incompletion costs. So the higher
the level of experience is, the more the cost of worker will be. How to allocate the workers to stations to obtain the best efficiency of the line and reduce the total worker cost is also a problem in ALB problems. This situation leads the multiobjective ALB problems. Multiobjective ALB problems become more complex because the decision maker need to make a trade-off between two or more conflicting objectives.

Determination of a set of optimal nondominated solutions effectively for multiobjective ALB considering workers’ capabilities is a big challenge for both academia and industry.

The problems of design and balancing of assembly lines have been extensively examined in the literature and a number of review studies have been published, including Tasan and Tunali [8], Levitin et al. [81], Rekiek et al. [23], Becker and Scholl [25] and Scholl and Becker [24]. Both exact and heuristic procedures-and more recently, meta-heuristic procedures-have been developed to solve this problem. Most of the papers assume that resources are homogeneous; hence, the duration of tasks does not depend on the stations to which they are assigned and any task can be carried out at any station. Working with heterogeneous resources, in terms of times and/or costs, involves solving a two-fold assignment problem: resources must be assigned to stations, while tasks are assigned simultaneously to those same stations. This case is often referred to as the assembly line design problem.

Pinto et al. [82] considered ALB problem where several alternatives are available in the choice of processing at each station. They described a method of simultaneously considering both the choice of manufacturing alternatives and the assignment of tasks to stations so as to minimize total costs (labor and fixed) over the expected life of the production line. Graves and Holmes [83] suggested an algorithm for assignment of activities and equipment to assembly line stations, satisfying the annual production rate. The objective of their work is to minimize the total worker cost that is composed of fixed equipment usage and tooling costs, variable equipment usage and set-up costs.

As set out in Becker and Scholl [25], the equipment selection problem is equivalent to a problem of selecting workers whose task performance speeds are different. However, in the works-see, for example, Akagi et al. [84], Wilson [85] and Lutz et al. [86]-it is normally assumed that the performance speed of all person who are manufacturing the same task is equal and that the time necessary to finish a task depends on the number of workers assigned to a station (on a linear basis with the number of workers in some papers and on a non-linear basis in others). In other cases, it is a question of deciding which workers should work on a given shift and which should not.

Hopp et al. [87] set out a case in which workers can vary in speed and are benchmarked by defining the speed factor of each worker relative to a “standard worker”; moreover, they assumed that a worker’s speed factor applies uniformly across all tasks. However, it is based on a line that is already designed and balanced, and the goal is to minimize the number of changes of workers from stations with surplus capacity to stations with a heavier workload, to help out temporarily.

From the above reviews, the previous research works lack consideration of allocating task and worker among stations according to different workers’ capabilities. It causes more idle time and reduction of the line efficiency. Moreover, most of the above previous researchers just concern with satisfying single objective. The improvement of line efficiency and reduction of total worker cost are incomparable or incommensurable and often competing objectives in this ALB problem.

The issue of this study is effective generation of a set of optimal nondominated so-
olutions with satisfying the minimization of cycle time, workload of station and worker cost concurrently for smALB problems considering workers’ capabilities.

For reducing the computation time, increasing the quality of obtained Pareto solutions, reducing the complexity of coding, a new approach is proposed to solve the multiobjective smALB problem considering workers’ capabilities.

3.2 Single-model Assembly Line Balancing Problem

As described in Chapter 1, the smALB produces one homogeneous product continuously in large quantities (mass production) (see Fig. 1.2).

As an illustrative example, an assembly line having 10 tasks, 4 stations and 4 workers is used. The processing time of worker and cost data are shown in Table 3.1. The precedence relationship graph is shown in Fig. 3.1.

Table 3.1: Data set of the smALB model

<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessor</th>
<th>Task time unit / Cost unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>j</td>
<td>Pre(j)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>{4}</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>{2,3}</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>{4}</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>{6}</td>
</tr>
<tr>
<td>8</td>
<td>{6}</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>{7,8,9}</td>
</tr>
<tr>
<td>10</td>
<td>{7,8,9}</td>
<td>26/74</td>
</tr>
</tbody>
</table>

Figure 3.1: Precedence relationship graph of the smALB model

From Table 3.1, workers have different processing time to tasks, different worker cost according to the different work experience.

The description of smALB problem considering workers’ capabilities is illustrated in Fig. 3.2.
Figure 3.2: Description of the smALB model

From Fig. 3.2, task sequence is generated according to precedence relationship of tasks. Based on trying to separate the task sequence into 4 parts, a breakpoint vector can be obtained. Using this breakpoint, the task sequence can be divided into 4 parts satisfying the station ordered. Worker allocation are generated randomly. According to above information, a solution (schedule) can be generated. Gantt chart and the corresponding schedule are shown at the bottom of Fig. 3.2.

The ALB problem concerns with the assignment of the tasks to stations and the allocation of the available workers for each station in order to minimize the cycle time, minimize the variation of workload and minimize the total worker cost under the constraint of precedence relationships.

In this study, the ALB problem subjects to the following assumptions:

A1. The precedence constraints among assembly tasks are known and constant.
A2. A worker is assigned to one station, and only processes the tasks assigned to that station.
A3. A task cannot be split amongst two or more stations.
A4. The processing time of worker for each task is known.
A5. Material handling, loading and unloading times, set-up and tool changing times are negligible, or are included in the processing times.
A6. Task processing time differs among workers because of workers’ differences in work experience.
A7. A worker can process all the tasks, and his/her work experience differs among tasks.
A8. Stations are located along a conveyor belt according to increased station index.
A9. Workers have different worker cost according to the different work experience.
3.3 Mathematical Formulation

Considering the workers’ capabilities of smALB problem, three objectives, minimization of the cycle time, minimization of the variation of workload and minimization of the total worker cost under the constraint of precedence relationship, are formulated. These three objectives are important in MS. Reducing cycle time means that can get higher producing efficiency in limited period. Decreasing variation of workload of machine can balance the usage of machine. Reducing the total worker cost means lower production cost.

The notation used in this Section can be summarized as follows.

**Notations**

**Indices**

\( i \): index of the stations \( (i = 1, \ldots, m) \).
\( j, k \): index of the tasks \( (j, k = 1, \ldots, n) \).
\( w \): index of the workers \( (w = 1, \ldots, m) \).

**Decision variables**

\[
\begin{align*}
x_{ij} &= \begin{cases} 
1, & \text{if task } j \text{ is assigned to station } i \\
0, & \text{otherwise}
\end{cases} \\
y_{iw} &= \begin{cases} 
1, & \text{if worker } w \text{ is working in station } i \\
0, & \text{otherwise}
\end{cases}
\]

**Parameters**

\( n \): number of tasks.
\( m \): number of stations/workers.
\( c_T \): cycle time of the assembly line.
\( d_{jw} \): worker cost of worker \( w \) process task \( j \).
\( d_T \): total worker cost.
\( t_{jw} \): processing time of task \( j \) by worker \( w \).
\( \text{Suc}(j) \): the set of direct successors of task \( j \).
\( \text{Pre}(j) \): the set of direct predecessors of task \( j \).
\( S_i \): the set of tasks assigned to station \( i \).
\( t(S_i) \): processing time at station \( i \). \( t(S_i) = \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jw} x_{ij} y_{iw}, \forall i \)
\( u_i \): utilization of the station \( i \). \( u_i = \frac{t(S_i)}{\max_{1 \leq i \leq m} \{t(S_i)\}} \)
\( \bar{u} \): average utilization of all stations. \( \bar{u} = \frac{1}{m} \sum_{i=1}^{m} u_i \)

**Mathematical Model**

The mathematical model for the smALB considering the workers’ capabilities can be stated as follows:
minimize \( \{ c_T, v, d_T \} \)

where

\[
c_T = \max_{1 \leq i \leq m} \left\{ \sum_{j=1}^{m} \sum_{w=1}^{m} t_{jw} x_{ij} y_{iw} \right\}
\]

\[
v = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (u_i - \bar{u})^2}
\]

\[
d_T = \sum_{i=1}^{m} \sum_{j \in S_i, w=1}^{m} d_{jw} y_{iw}
\]

s.t.

\[
\sum_{i=1}^{m} t_{ij} \geq \sum_{i=1}^{m} t_{i,k}, \quad \forall k \in Pre(j), \forall j
\]

\[
\sum_{i=1}^{m} x_{ij} = 1, \forall j
\]

\[
\sum_{w=1}^{m} y_{iw} = 1, \forall i
\]

\[
\sum_{i=1}^{m} y_{iw} = 1, \forall w
\]

\[
x_{ij} \in \{0, 1\} \forall i, j
\]

\[
y_{iw} \in \{0, 1\} \forall i, w
\]

The first objective (Eq. 3.1) of the model is to minimize the cycle time of the assembly line. The second objective (Eq. 3.2) is to minimize the variation of workload. The third objective (Eq. 3.3) is to minimize the total worker cost. Inequity (3.4) states that all predecessors of task \( j \) must be assigned to a station, which is in front of or the same as the station that task \( j \) is assigned in. Equation (3.5) ensures that task \( j \) must be assigned to only one station. Equation (3.6) ensures that only one worker can be allocated to station \( i \). Equation (3.7) ensures that worker \( w \) can be allocated to only one station. Constraints (3.8) and (3.9) represent the usual integrity restrictions.

### 3.4 Multiobjective Genetic Algorithm Approach

#### 3.4.1 Main Ideas

Because the ALB problem in MS belong to classical combinatorial optimization problem. This problem has been proved which falls into the NP-hard class of problem. Moreover, multiobjective optimization makes this problem more difficult and complex. The traditional optimum seeking methods have been proved not to be of practical use for large problems due to their computational inefficiency. Evolutionary algorithms seem to be especially suited to multiobjective optimization because they are able to explore multiple Pareto-optimal solutions in a single simulation run and may exploit space of solutions by recombination. Genetic algorithms as one of evolutionary algorithms is used to resolve the optimization of multiple objectives simultaneously.

In this study, a new approach is proposed to solve the multiobjective smALB problem by simplifying the complexity of coding, reducing the computation time, and
improving the quality of solutions with satisfying the minimization of cycle time, workload of station and worker cost concurrently for smALB problems considering workers’ capabilities.

For reducing the complexity of coding of this problem, a random key-based coding method considering the specific chromosome structure and the characteristics of the problem is adopted. Use this coding method can generate feasible solutions easily so as to avoid doing unnecessary process for infeasible solutions.

Moreover, this real number based coding can easy apply arithmetic crossover processing to improve the abilities of exploration and exploitation of method.

As one part of proposed approach, awGA (as introduced in Section 2.2.1) utilizes some useful information from current population to readjust weights in order to obtain a search pressure towards to positive ideal point. It improves the ability of convergence of solutions and increases the quality of solutions.

Moreover, awGA uses a simple method to normalize the multiple objectives into one value. This mechanism can reduce the computation time.

In this section, a genetic algorithm approach using random key-based encoding method considering the specific chromosome structure and the characteristics of the problem is proposed. In this approach, a combination of the arithmetic crossover and weight mapped crossover (WMX), swap mutation and immigration operator were used as genetic operations, and the adaptive weight approach (AWA) was used to determine fitness value.

The overall structure of adopted random key-based GA is given as follows:

**procedure**: rkGA for smALB problem  
**input**: data set of smALB problem, GA parameters (popSize, maxGen, pM, pC, pI)  
**output**: Pareto optimal solutions $E$

begin  
$t ← 0$;  
initialize $P(t)$ by random key-based encoding routine;  
calculate objectives $c_T(P), v(P), d_T(P)$ by random key-based decoding routine;  
create Pareto set $E(P)$;  
calculate fitness $eval(P)$ by adaptive-weight approach;  
while (not terminating condition) do  
create $C(t)$ from $P(t)$ by arithmetic and weight mapping crossover routine;  
create $C(t)$ from $P(t)$ by swap mutation routine;  
create $C(t)$ from $P(t)$ by immigration routine;  
calculate objectives $c_T(C), v(C), d_T(C)$ by random key-based decoding routine;  
update Pareto set $E(P)$;  
calculate fitness $eval(P, C)$ by adaptive-weight approach;  
select $P(t + 1)$ from $P(t)$ and $C(t)$ by roulette wheel selection routine;  
$t ← t + 1$;  
end  
**output** Pareto optimal solutions $E(P)$
end
3.4.2 Genetic Representation

In ALB problem, encoding a solution into a chromosome is a key issue for GAs. For this issue, the priority-based genetic algorithm (priGA), which was proposed by Gen and Cheng [88] are modified to solve shortest path problems. A chromosome representation method of random key-based genetic algorithm (rkGA) with multiple vectors, i.e. task priority vector (real number code), worker allocation vector (integral number code), task sequence vector (integral number code) and breakpoint vector (integral number code) is proposed.

The detailed encoding and decoding process of a chromosome consists of four phases:

Phase 1: Creating a task sequence
  - step 1.1: Generating a random key-based chromosome by encoding method.
  - step 1.2: Decoding a feasible task sequence $T_5$ that satisfies the precedence constraint

Phase 2: Assigning workers to each station
  - step 2.1: Encoding for worker allocation by randomly assign the worker to each station.

Phase 3: Assigning tasks to each station
  - step 3.1: Breakpoint decoding to assign tasks to stations.

Phase 4: Designing a schedule
  - step 4.1: Creating a schedule for the assembly line.

While introducing these three phases, in order to illustrate the procedures for smALB, an assembly line having 10 tasks, 4 stations and 4 worker is used. The data set is given in Table. 3.1 and its corresponding precedence relationship graph given in Fig. 3.1.

Phase 1: Creating a task sequence

step 1.1: Generating a random key-based chromosome by encoding method.

The chromosome is composed of two parts (see Fig. 3.3); the first part is the task priority vector and the second part is worker allocation vector. In this study, the task

<table>
<thead>
<tr>
<th>Locus: task id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task priority ($v_1$)</td>
<td>3.31</td>
<td>7.14</td>
<td>6.59</td>
<td>2.24</td>
<td>8.67</td>
<td>5.38</td>
<td>4.13</td>
<td>9.87</td>
<td>10.72</td>
<td>1.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Locus: station id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worker allocation ($v_2$)</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3.3: Task priority and worker allocation vector of the example for smALB

priority vector is presented as a random key-based encoding, i.e., the locus of each gene represent task ID and gene are presented as real number. The locus of each gene
in worker allocation vector represent station ID and gene are presented as worker ID. This gene is generated by randomly. An example of generated random key-based chromosome is illustrated in Fig. 3.3 For the task priority vector, random key-based encoding method to generate priority is adopted. For task 7, the generated priority is 4.13 (as shown in Fig. 3.3).

For the worker allocation vector, normal way is employed to produce the allocation such as randomly encoding. For station 3, the corresponding worker allocation is 1 (as shown in Fig. 3.3), which means the worker 1 will be assigned to station 3 to the work.

The procedure of encoding for task priority vector and worker allocation vector are shown in Fig. 3.4 and Fig. 3.5.

---

**procedure**: encoding for task priority \( v_1 \)

**input**: \( n \): total number of tasks

**output**: \( v_1 \): task priority

**begin**

for \( i = 1 \) to \( n \)

\( v_1[i] \leftarrow \text{random}[i:i+1] \)

for \( i = 1 \) to \( \lceil n/2 \rceil \)

repeat

\( l \leftarrow \text{random}[1, n] \)

\( j \leftarrow \text{random}[1, n] \)

until \( l \neq j \)

swap \( (v_1[l], v_1[j]) \)

output task priority \( v_1 \)

**end**

---

Figure 3.4: Procedure of encoding for task priority vector of smALB

**step 1.2**: Decoding a feasible task sequence \( T_5 \) that satisfies the precedence constraint.

Creating a task sequence procedure is same as priority-based decoding procedure proposed by Gen and Cheng [88]. At the beginning, try to find a task for the first station. Task 1, 2, 3, 7, 9 are eligible for the position, which can be easily fixed according to adjacent relation among tasks. From the task priority vector, priorities of them are 3.31, 7.14, 6.59, 4.13, and 10.72 respectively. The task 9 with the highest priority is put into the task sequence. Then delete task 9 off the precedence relationship graph. Next, Task 1, 2, 3, 7 are eligible for the next position. Because node 2 has the largest priority value, it is put into the task sequence. The produce of this decoding a task sequence is shown in Fig. 3.6. The trace table for this procedure is shown in Table. 3.2 By the same manner, the chromosome can easily be encoded. The generated task sequence vector is shown in Fig. 3.7.

**Phase 2: Assigning worker to each station**

**step 2.1**: Encoding for worker allocation by randomly assign the worker to each station.
Multiobjective smALB Considering Workers’ Capabilities

**Figure 3.5: Procedure of encoding for worker allocation vector of smALB**

```
procedure : encoding for worker allocation v₂
input: m: number of tasks
output: v₂: worker allocation
begin
    for i=1 to m
        v₂[i] ← i
    end
    for i=1 to [m/2]
        repeat
            l ← random[1, m];
            j ← random[1, m];
            until l ≠ j
            swap (v₂[l], v₂[j]);
        end
    output worker allocation v₁
end
```

**Figure 3.6: Procedure of decoding of task sequence for smALB**

```
procedure : decoding for task sequence v₃
input: n: total no. of tasks, v₁: task priority
        Pre(i): set of predecessors
output: task sequence v₃
begin
    v₃ ← Ø;
    A ← {i | Pre(i)=Ø} // A is the available task set
    while |A|>0 do
        j ← arg max(v₁[i]); // find the available task with highest priority
        v₃ ∪ {j}; // update the task sequence
        A ← A \ {j}; // update the available task set
        A ← A U {i | i ∉ v₃ and Pre(i)⊆ v₃};
    end;
    output task sequence v₃;
end
```

**Figure 3.7: Task sequence vector of the example for smALB**

```
Task sequence (v₃)  1  2  3  4  5  6  7  8  9 10
                  9  2  3  5  7  1  4  6  8 10
```
Table 3.2: Trace table of decoding of task sequence for smALB

<table>
<thead>
<tr>
<th>Iteration</th>
<th>A</th>
<th>j</th>
<th>v_j</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(1,2,3,7,9)</td>
<td>9</td>
<td>[9]</td>
</tr>
<tr>
<td>1</td>
<td>(1,2,3,7)</td>
<td>2</td>
<td>[9 2]</td>
</tr>
<tr>
<td>2</td>
<td>(1,3,7)</td>
<td>3</td>
<td>[9 2 3]</td>
</tr>
<tr>
<td>3</td>
<td>(1,5,7)</td>
<td>5</td>
<td>[9 2 3 5]</td>
</tr>
<tr>
<td>4</td>
<td>(1,7)</td>
<td>7</td>
<td>[9 2 3 5 7]</td>
</tr>
<tr>
<td>5</td>
<td>(1)</td>
<td>1</td>
<td>[9 2 3 5 7 1]</td>
</tr>
<tr>
<td>6</td>
<td>(4)</td>
<td>4</td>
<td>[9 2 3 5 7 1 4]</td>
</tr>
<tr>
<td>7</td>
<td>(6)</td>
<td>6</td>
<td>[9 2 3 5 7 1 4 6]</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>8</td>
<td>[9 2 3 5 7 1 4 6 8]</td>
</tr>
<tr>
<td>9</td>
<td>(10)</td>
<td>10</td>
<td>[9 2 3 5 7 1 4 6 8 10]</td>
</tr>
</tbody>
</table>

In this phase, the worker allocation vector ($v_2$) is coded by randomly as shown in Fig. 3.3, which indicates the worker assigned for each station. The locus of worker allocation vector represents the station ID.

Phase 3: Assigning tasks to each station

step 3.1: Breakpoint decoding to assign tasks to stations.

The random key-based decoding procedure consists of three main steps:

The procedure of decoding for breakpoint vector

Step 1: Calculate the lower bound ($c_{LB}$) and the upper bound ($c_{UB}$) of the cycle time for the solution represented by the task sequence vector and worker allocation vector.

$$c_{LB} = \left\lceil \frac{1}{m} \sum_{j=1}^{n} \min_{1 \leq w \leq m} \left\{ t_{jw} \right\} \right\rceil$$

$$c_{UB} = \left\lceil \frac{1}{m} \sum_{j=1}^{n} \max_{1 \leq w \leq m} \left\{ t_{jw} \right\} \right\rceil + \max_{1 \leq j \leq n} \left\{ t_{jw} \right\}$$

Step 2: Find out optimal cycle time by bisection searching.

Step 3: Partition the task sequence into $m$ parts with the optimal cycle time based on the worker allocation vector.

The produce of this decoding a breakpoint vector is shown in Fig. 3.8. The trace table for this procedure is shown in Table. 3.3. After applying the decoding procedure to the example, the breakpoint vector in Fig. 3.9 is obtained.

In this phase, this worker allocation vector ($v_2$) will decode into a solution with the task sequence vector ($v_3$), simultaneously by the breakpoint decoding procedure. It
procedure: decoding for breakpoint vector \( v_4 \)

**input:** 
- \( m \): total no. of stations, 
- \( n \): total no. of tasks, 
- \( v_4[k] \): worker allocation, 
- \( v_3[j] \): task sequence, 
- \( c_{lb} \): low bound of cycle time, 
- \( c_{ub} \): upper bound of cycle time, 
- \( t[j] \): processing time

**output:** 
- \( v_4 \): breakpoint vector

```
begin
while \((c_{ub} - c_{lb} > 1)\) do
    \( c_T \leftarrow (c_{lb} + c_{ub})/2; \)
    \( i \leftarrow 1; w_t \leftarrow 0; S_i \leftarrow \emptyset; \)
    for \( j = 1 \) to \( n \) do
        \( k \leftarrow v_3[j]; w \leftarrow v_2[i]; \)
        if \( w_t + t[k][w] > c_T \) then
            \( i \leftarrow i + 1; w_t \leftarrow 0; S_i \leftarrow \emptyset; \)
        if \( \geq m \) then break:
            \( w \leftarrow v_2[i]; \)
            \( w_t \leftarrow w_t + t[k][w]; S_i \leftarrow S_i \cup \{k\}; v_4[i] \leftarrow j; \)
        if \( i \leq m \) then
            \( c_{lb} \leftarrow c_T; // update upper bound and lower bound \)
        else
            \( c_{ub} \leftarrow c_T; \)
    output breakpoint vector \( v_4 \);
end
```

Figure 3.8: Procedure of decoding of breakpoint for smALB

**Table 3.3: Trace table of decoding of breakpoint for smALB**

<table>
<thead>
<tr>
<th>No.</th>
<th>( c_{lb} )</th>
<th>( c_{ub} )</th>
<th>( c_T )</th>
<th>( i )</th>
<th>( v_4[j] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
<td>120</td>
<td>83</td>
<td>4</td>
<td>[1 4 7 9]</td>
</tr>
<tr>
<td>2</td>
<td>46</td>
<td>83</td>
<td>65</td>
<td>5</td>
<td>[1 4 6 8 10]</td>
</tr>
<tr>
<td>3</td>
<td>65</td>
<td>83</td>
<td>74</td>
<td>4</td>
<td>[1 4 7 9]</td>
</tr>
<tr>
<td>4</td>
<td>65</td>
<td>74</td>
<td>70</td>
<td>4</td>
<td>[1 4 7 9]</td>
</tr>
<tr>
<td>5</td>
<td>65</td>
<td>70</td>
<td>68</td>
<td>5</td>
<td>[1 4 6 8 10]</td>
</tr>
<tr>
<td>6</td>
<td>68</td>
<td>70</td>
<td>69</td>
<td>4</td>
<td>[1 4 7 9]</td>
</tr>
<tr>
<td>7</td>
<td>68</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Breakpoint vector (\( v_4 \))

```
  1  4  7  9
```

Figure 3.9: Breakpoint vector of the example for smALB
means that when the tasks are assigned into each station, the worker will be assigned into each station at the same time. Based on the breakpoint vector, the task sequence vector is separated into 4 parts, all of which respectively correlated with one station. According to worker allocation vector, the workers can be assigned to each station. Fig. 3.10 illustrates the phenotype of the example chromosome.

**Phase 4: Designing a schedule**

**step 4.1:** Creating a schedule for the assembly line.

Using the chromosomes, the schedule can be constructed as follows:

<table>
<thead>
<tr>
<th>Schedule</th>
<th>$S = (j_i, t_i, w_i, t_{j_{ij}})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S = {(9, 1, 3, 0-17), (2, 1, 3, 17-33),$</td>
<td></td>
</tr>
<tr>
<td>$(3, 1, 3, 33-60), (5, 2, 4, 60-82),$</td>
<td></td>
</tr>
<tr>
<td>$(7, 2, 4, 82-116), (1, 2, 4, 116-129),$</td>
<td></td>
</tr>
<tr>
<td>$(4, 3, 1, 129-158), (6, 3, 1, 158-186),$</td>
<td></td>
</tr>
<tr>
<td>$(8, 4, 2, 186-219), (10, 4, 2, 219-246)}$</td>
<td></td>
</tr>
</tbody>
</table>

### 3.4.3 Genetic Operators

Genetic operators mimic the process of heredity of genes to create new offspring at each generation. Using the different genetic operators has very large influence on GA performance. Therefore, it is important to examine different genetic operators. Crossover is the main genetic operator. It operates on two parents (chromosomes) at a time and generates offspring by combining both chromosomes’ features.
Crossover operator

In this study, arithmetic crossover is used for task priority vector and weight mapped crossover (WMX) is used for worker allocation vector. Notation $p_C$ are defined as crossover probability.

**Arithmetic Crossover (for task priority vector)**

The basic concept of this kind of operator is borrowed from the convex set theory. Generally, the weighted average of two vectors $v_j(i)$ ($i$-th gene) and $v_k(i)$ of $j$-th chromosome and $k$-th chromosome, is calculated as follows:

$$\lambda_1 v_j(i) + \lambda_2 v_k(i)$$

(3.12)

if the multipliers are restricted as

$$\lambda_1 + \lambda_2 = 1, \lambda_1 > 0 \text{ and } \lambda_2 > 0$$

(3.13)

The multipliers $\lambda_1$ and $\lambda_2$ are randomly generated and satisfied Equation. 3.13

The weighted form is known as convex combination. If the nonnegative condition on the multipliers is dropped, the combination is known as an affine combination. Finally, if the multipliers are simply required to be in real space $E$, the combination is known as a linear combination. Similarly, arithmetic operators are defined as the combination of two chromosomes $v_j$ and $v_k$ as follows:

$$v_j'(i) = \lambda_1 v_j(i) + \lambda_2 v_k(i), \forall i$$

(3.14)

$$v_k'(i) = \lambda_1 v_k(i) + \lambda_2 v_j(i), \forall i$$

(3.15)

The example of arithmetic crossover for task priority is shown in Fig. 3.11.

<table>
<thead>
<tr>
<th>Locus: task id</th>
<th>Task priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent $x_1$:</td>
<td>3.31 7.14 6.59 2.24 8.67 5.38 4.13 9.87 10.72 1.91</td>
</tr>
<tr>
<td>parent $x_2$:</td>
<td>7.19 9.86 1.35 4.26 3.77 8.61 10.53 2.46 5.32 6.99</td>
</tr>
<tr>
<td>offspring $x_1'$:</td>
<td>4.47 7.96 5.02 2.85 7.20 6.35 6.05 7.65 9.10 3.43</td>
</tr>
<tr>
<td>offspring $x_2'$:</td>
<td>6.03 9.88 2.92 3.65 2.60 7.64 8.61 4.68 5.96 5.47</td>
</tr>
</tbody>
</table>

Figure 3.11: Illustration of arithmetic crossover for task priority vector for smALB

**Weight Mapped Crossover (WMX) (for worker allocation vector)**

WMX can be viewed as an extension of one-point crossover for permutation representation. As one-point crossover, (1) two chromosomes (parents) would be to choose a random cut-point, (2) generate the offspring by using segment of own parent to the left of the one-cut point, (3) and then, remapping the right segment that base on the weight of other parent of right segment. The procedure of WMX is shown in Fig. 3.12

The example of WMX for worker allocation is shown in Fig. 3.13.
procedure weight mapping crossover
input : \( v_1, v_2 \); worker allocation vector
\( n \); number of worker station
output : \( v'_1, v'_2 \)
begin
\( p \leftarrow \text{random}[1, n]; \) \( l \leftarrow n - p; \)
for \( i = 1 \) to \( l \)
\( s_1[i] \leftarrow v_1[p+i]; \) \( s_2[i] \leftarrow v_2[p+i]; \)
\( s_i[l] \leftarrow \text{sorting}(s_i[l]); \) \( s_i[l] \leftarrow \text{sorting}(s_i[l]); \)
\( v'_1[p+i] \leftarrow v_1[p+i]; \) \( v'_2[p+i] \leftarrow v_2[p+i]; \)
end

Figure 3.12: Procedure of weight mapped crossover for worker allocation vector for smALB

Figure 3.13: Illustration of weight mapped crossover for worker allocation vector for smALB
Mutation operator

Mutation is a background operator, which produces spontaneous random changes in various chromosomes. A simple way to achieve mutation would be to alter one or more genes. In GAs, mutation serves the crucial role of either replacing the genes lost from the population during the selection process so that they can be tried in a new context or providing the genes that were not present in the initial population. In this research, it is relatively easy to produce some mutation operators for the random-key based genetic representation.

Several mutation operators have been proposed for the rkGA representation, such as swap mutation, inversion mutation, and insertion mutation, and so on. In this study, swap mutation is used for generating various offspring for task priority vector and worker allocation vector. Swap mutation selects two positions at random and then swaps the gene on these positions. Notation $p_M$ is defined as mutation probability. The example of swap mutation is shown in Fig. 3.14.

Immigration operator

The trade-off between exploration and exploitation in serial GAs for function optimization is a fundamental issue. To search effectively and efficiently, a GA must maintain a balance between these two opposing forces. Moed et al. [89] proposed an immigration operator which, for certain types of functions, allows increased exploration while maintaining nearly the same level of exploitation for the given population size. The algorithm is modified to the following four steps; (1) apply immigration routine in each generation, (2) generate and (3) evaluate new $\text{popSize} \cdot p_I$ random individuals, and (4) replace the $\text{popSize} \cdot p_I$ worst individuals of the population with the new generated $\text{popSize} \cdot p_I$ random individuals ($p_I$, called the immigration probability).

Selection operator

Selection (reproduction) operator is intended to improve the average quality of the population by giving the high quality chromosomes a better chance to get copied into the next generation. In this study, roulette wheel selection (RWS) is used to process selection operator. It belongs to the fitness-proportional selection and can select a new population with respect to the probability distribution based on fitness values. The selection process is based on spinning the wheel the number of times equal to population size, each time selection a single chromosome for the new population. The wheel
features the selection method as a stochastic sampling procedure. The procedure of RWS is shown in Fig. 3.15, $v_k$ is the $k$-th chromosome.

---

**Figure 3.15: Procedure of roulette wheel selection for smALB**

### 3.4.4 Fitness Function

While considering multi-objective problem, a key issue is to determine the weight of each objective. Gen and Cheng [27] proposed an Adaptive Weight Approach (AWA) that utilizes some useful information from the current population to readjust weights for obtaining a search pressure toward a positive ideal point. In this study, the following objectives are used:

1. minimization of the cycle time ($c_T$).
2. minimization of variation of workload ($v$).
3. minimization of total worker cost ($d_T$).

For the minimization case, transforming the original problem into its equivalent maximization problem is needed firstly, and then use AWA method.

$$\max \{ z_1 = \frac{1}{c_T}, z_2 = \frac{1}{v}, z_3 = \frac{1}{d_T} \} \quad (3.16)$$

For the solutions at each generation, $z_{q}^{\text{max}}$ and $z_{q}^{\text{min}}$ are the maximal and minimal values for the $q$-th objective as defined by the following equations:

$$z_{q}^{\text{max}} = \max \{ f_q(v_k), \; k = 1, 2, ..., \text{popSize} \}, q = 1, 2, 3 \quad (3.17)$$

$$z_{q}^{\text{min}} = \min \{ f_q(v_k), \; k = 1, 2, ..., \text{popSize} \}, q = 1, 2, 3 \quad (3.18)$$

The adaptive weights are calculated as:
The weighted-sum objective function for a given chromosome is then given by the following equation:
\[
eval(v_k) = \sum_{q=1}^{3} w_q (f_q(v_k) - z_{q}^{\text{min}}), k = 1, 2, \ldots, \text{popSize}
\] (3.20)

3.5 Experiments and Discussion

In order to illustrate the applicability of the proposed method, first try to solve the example problem in Fig. 3.1. A weight factor method was used to get a best compromise solution in Pareto set. The weight factors for cycle time objective, variation of workload objective, total worker cost objective were set to \(w_1 = 0.5, w_2 = 0.2\) and \(w_3 = 0.3\), respectively. The best compromised solution is shown in Fig. 3.16.

In the real-world, the assembly line is not just for producing one unit of the product. It should produce several units. Fig. 3.17 show the Gantt charts for three units of product. Gantt chart and schedule of the best compromised solution for weight \((w_1=1, w_2=0, w_3=0)\), \((w_1=0, w_2=1, w_3=0)\) and \((w_1=0, w_2=0, w_3=1)\) are shown in Fig. A.1, A.2 and A.3.

Later, in order to evaluate the performance of the proposed method, a large set of problems were used. Since in the literature, no benchmark data sets exist for this problem, Gunther’s problem data from web are employed, which is widely used in the ALB problem literature [90], as the base data of the problem (see Fig. A.4 and Table A.1) and added more data (see Table A.2) to have 35 tasks, 6-15 stations and 6-15
Experiments and Discussion

Figure 3.17: Gantt chart of the best compromise solution (3 products) with $w_1=0.5$, $w_2=0.2$ and, $w_3=0.3$ of smALB

workers in the assembly line. For each problem, the number of station is equal to the number of workers, and each task can be processed on any worker. The worker cost data are generated at random according to worker’s capability [97]. The parameters are set following:

- Population size, $popSize=100$;
- Maximum generations, $maxGen=500$;
- Crossover probability, $p_c=0.60$;
- Mutation probability, $p_M=0.20$;
- Immigration probability, $p_I=0.06$.

The program was written in C# language, and implemented on HP-dx2100 PC, with a 2.8G Intel Pentium-4 CPU, and 1024 MB memory. For this problem, the procedure need to be run with 20 times and compared with the results of rkGA and priGA (different priority key-based coding, order based crossover and weight mapping crossover from rkGA).

Let $S_j$ be a solution set for each solution method ($j=1, 2$). In this study, the following two performance measures are considered. They provide a good estimate of convergence if a reference set for $PF^*$ (i.e., the Pareto optimal solution set or a near Pareto optimal solution set) is chosen [4]. Because no researches have published their Pareto solutions for this problem, therefore, The reference Pareto solutions $PF^*$ come from combining the obtained Pareto solutions by the two methods with special bigger parameter settings than above pre-defined parameters.

**Generational distance** $GD_R(S_j)$

The generational distance (average distance) measure finds an average closest distance of the solutions of $S_j$ from $PF^*$ (see page. [44] for details).

The comparison results of two solution sets using the coverage are depicted using Box-and-whisker plots. Box-and-whisker plots are helpful in interpreting the distribution of data. The median of a set of data separates the data into two equal parts. Data can be further separated into quartiles.

- The first quartile is the median of the lower part of the data.
- The second quartile is another name for the median of the entire set of data.
- The third quartile is the median of the upper part of the data.
Figure 3.18: Comparison of the generational distance about priGA and rkGA for smALB

Table 3.4: Comparison of the generational distance (mean and standard deviation) about priGA and rkGA for smALB

<table>
<thead>
<tr>
<th></th>
<th>Mean GD(rkGA)</th>
<th>Mean GD(priGA)</th>
<th>Reduced ((GD(rkGA)−GD(priGA))/GD(priGA))</th>
<th>Standard Deviation GD(rkGA)</th>
<th>Standard Deviation GD(priGA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>35(6)</td>
<td>0.5643</td>
<td>0.5980</td>
<td>-5.63%</td>
<td>0.2205</td>
<td>0.1693</td>
</tr>
<tr>
<td>35(7)</td>
<td>0.6223</td>
<td>0.6117</td>
<td>1.74%</td>
<td>0.1645</td>
<td>0.1418</td>
</tr>
<tr>
<td>35(8)</td>
<td>0.6353</td>
<td>0.6433</td>
<td>-1.24%</td>
<td>0.2426</td>
<td>0.2206</td>
</tr>
<tr>
<td>35(9)</td>
<td>0.4480</td>
<td>0.4747</td>
<td>-5.62%</td>
<td>0.1556</td>
<td>0.1828</td>
</tr>
<tr>
<td>35(10)</td>
<td>1.1880</td>
<td>1.2840</td>
<td>-7.48%</td>
<td>0.3463</td>
<td>0.4682</td>
</tr>
<tr>
<td>35(11)</td>
<td>0.8743</td>
<td>0.8580</td>
<td>1.90%</td>
<td>0.3174</td>
<td>0.3644</td>
</tr>
<tr>
<td>35(12)</td>
<td>0.8997</td>
<td>0.9360</td>
<td>-3.88%</td>
<td>0.3374</td>
<td>0.3262</td>
</tr>
<tr>
<td>35(13)</td>
<td>1.1957</td>
<td>1.1950</td>
<td>0.06%</td>
<td>0.3411</td>
<td>0.3164</td>
</tr>
<tr>
<td>35(14)</td>
<td>0.6087</td>
<td>0.7227</td>
<td>-15.77%</td>
<td>0.2086</td>
<td>0.2167</td>
</tr>
<tr>
<td>35(15)</td>
<td>0.7717</td>
<td>0.9243</td>
<td>-16.52%</td>
<td>0.2114</td>
<td>0.3058</td>
</tr>
</tbody>
</table>

-5.24%
• Quartiles separate the original set of data into four equal parts.

Each of these parts contains one-fourth of the data.

Comparing average distance (as shown in Fig. 3.18 and Table 3.4) of the two ways, rkGA is slightly better than priGA (about 5% improved in convergence performance).

The Fig. 3.19 and Table 3.5 show the comparison of CPU times of each approaches. It is easy to see that the rkGA are faster than priGA (about 48% reduced in computation time).

![CPU times (mean) of rkGA](image1)

![CPU times (mean) of priGA](image2)

Figure 3.19: Comparison of means of CPU times by rkGA and priGA of smALB

Table 3.5: Comparison of CPU time (mean and standard deviation) about priGA and rkGA for smALB

<table>
<thead>
<tr>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU(rkGA)</td>
<td>CPU(priGA)</td>
</tr>
<tr>
<td>35(6)</td>
<td>40.2147</td>
</tr>
<tr>
<td>35(7)</td>
<td>25.1080</td>
</tr>
<tr>
<td>35(8)</td>
<td>20.6817</td>
</tr>
<tr>
<td>35(9)</td>
<td>20.3917</td>
</tr>
<tr>
<td>35(10)</td>
<td>19.8563</td>
</tr>
<tr>
<td>35(11)</td>
<td>20.3480</td>
</tr>
<tr>
<td>35(12)</td>
<td>20.0903</td>
</tr>
<tr>
<td>35(13)</td>
<td>20.2687</td>
</tr>
<tr>
<td>35(14)</td>
<td>20.4080</td>
</tr>
<tr>
<td>35(15)</td>
<td>20.0723</td>
</tr>
</tbody>
</table>

The results indicate that the proposed approach could generate a set of optimal nondominated solutions effectively satisfying the multiple objectives concurrently for smALB problems with considering workers' capabilities. Comparing rkGA to priority key-based genetic algorithm (priGA) (different integer priority key-based coding, one cut point crossover from rkGA) indicates that rkGA could achieve both better efficacy
(about 5% improved in convergence performance) and efficiency (about 48% reduced in computation time) performances than priGA.

3.6 Summary

In this Chapter, for solving the multiobjective smALB problem considering workers’ capabilities, the mathematical models for minimization of cycle time, variation of workload of station, and total worker cost are extracted firstly.

For simplifying the complexity of coding, reducing the computation time, and improving the quality of solutions for multiobjective smALB problem considering different workers’ capabilities, a new multiobjective genetic algorithm is proposed.

Random key-based coding method could reduce the complexity of coding of this problem. In addition, applying genetic operators on this coding method could improve the abilities of exploration and exploitation.

Adaptive weights method could improve the ability of convergence of solutions and improve the quality of solutions, and normalization of multiple objectives into one value could reduce the computation time.
Chapter 4

Multiobjective mALB with Novel Cycle Time Reference

This Chapter is organized as follows: Section 4.1 introduces the background and related research works of problem. Section 4.2 describes the problem and assumptions. The mathematical formulations for this problem are extracted in Section 4.3. Section 4.4 introduces the details of approach to resolve this problem and experiments and discussion are addressed in Section 4.5. Last Section 4.6 summaries this Chapter.

4.1 Introduction

Different from the descriptions of smALB problem (section 3.1), mALB process a set of similar models of a product, e.g., versions of a car, which may differ from each other with respect to size, color etc. Considering the workers’ capabilities, multiobjective mALB problem concerns with the assignment of the proper worker to the proper station and assigning the suitable task of different models to the suitable station in order to generate a set of optimal solutions (schedules) rapidly and effectively.

As an extension of smALB, mALB with multiple models not only need to consider the allocation of tasks and workers among stations, but also need to consider the sequence and amount of models. It makes ALB problem different from traditional smALB.

Different workers’ capabilities lead the mALB problem more complex to allocate manufacturing resources and human resources. The higher the level of experience is, the more the cost of worker will be. How to allocate the workers to stations to obtain the best efficiency of the line and reduce the total worker cost is also a big problem in mALB problems. Consideration of balancing the cycle time and worker cost make this problem more difficult.

Traditional method often combine the multiple models into one model according to the precedence relationships of different models firstly. Then, use a reference cycle time such as average cycle time, minimum or maximum cycle time of models to separate the task sequences.

However, as reviewed recent papers at Section 1.2.3 and Section 3.1, the previous researchers do not considers that the cycle time need associated with the demand ratio of each model in mALB problem. The average cycle time of different models can not reflect that demand ratio of different models impact on cycle time. So, the demand-
based cycle time is become more important. Moreover, there is a lack of papers, which have studied mALB problems considering the different workers’ capabilities to optimize the three objectives: minimization of cycle time, minimization of variation of workload and minimization of the total worker cost simultaneously.

As introduced in Section 1.2 and 3.1 moEA methods are usually used to solve this multiobjective ALB problems, such as awGA, which introduced in Chapter 3. However, some methods can not get better convergence.

The issue of this study is effective generation of a set of optimal nondominated solutions with satisfying the minimization of cycle time, workload of station and worker cost concurrently for mALB problems considering workers’ capabilities.

For reducing the computation time, increasing the quality of obtained Pareto solutions, reducing the complexity of coding, a new approach is proposed to solve the multiobjective mALB problem considering workers’ capabilities.

### 4.2 Mixed-model Assembly Line Balancing Problem

With this increasing and varied demands of customers, manufacturers are forced to produce different models of the same product on an assembly line so that potential market is not lost. In such environments, mAL appear to be the most appropriate. In a mAL, a set of similar models of a product, e.g., versions of a car, which may differ from each other with respect to size, color etc. can be assembled simultaneously. Nowadays, they are often adopted in the automobile industry, e.g. the German car manufacturer BMW offers a catalogue of optional features which, theoretically, results in $10^{32}$ different models [92], and similar industries in order to avoid unnecessary inventories and increase manufacturing flexibility to respond to the ever changing demands of the customers.

Most research address two key problems in the design of mALs, i.e., the mALB which involves the assignment of tasks to stations and mixed-model sequencing which involves the sequencing of models on the line [93]. In this Chapter, the mALB model without sequencing is considered.

As an illustrative example, an assembly line having 10 tasks, 4 stations, 4 worker and 2 models is used.

The description of mALB problem considering workers’ capabilities is illustrated in Fig. 4.1. From Fig. 4.1 the precedence relationships of two models are combined into one model. Task sequence is generated according to combined precedence relationship of tasks. Based on trying to separate the task sequence into 4 parts, a breakpoint vector can be obtained. Using this breakpoint, the task sequence can be divided into 4 parts satisfying the station ordered. Worker allocation are generated randomly. According to above information, a solution (schedule) can be generated. Gantt chart of two models and the corresponding schedule are shown at the bottom of Fig. 3.2.

The precedence relationship graph and the processing time of worker for model 1 are shown in Fig. 4.2 and Table 4.1. The precedence graph and the processing time of worker for model 2 are shown in Fig. 4.3 and Table 4.2. The data set of cost is shown in Table 4.3.
Mixed-model Assembly Line Balancing Problem

Figure 4.1: Description of the mALB model

Figure 4.2: Precedence relationship graph of model 1 for the mALB
Table 4.1: Data set of model 1 for mALB

<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessors</th>
<th>Task time unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>j</td>
<td>Pre(j)</td>
</tr>
<tr>
<td>1</td>
<td>Ø</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Ø</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>{3}</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>{1}</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>Ø</td>
<td>42</td>
</tr>
<tr>
<td>8</td>
<td>{6}</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>Ø</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>{7,8,9}</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure 4.3: Precedence relationship graph of model 2 for the mALB

Table 4.2: Data set of model 2 for mALB

<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessors</th>
<th>Task time unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>j</td>
<td>Pre(j)</td>
</tr>
<tr>
<td>1</td>
<td>Ø</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>Ø</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>Ø</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>{2,3}</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>{4}</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>Ø</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>{8,9}</td>
<td>26</td>
</tr>
</tbody>
</table>
Table 4.3: Data set of cost for mALB

<table>
<thead>
<tr>
<th>Task</th>
<th>Cost unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$w_1$</td>
</tr>
<tr>
<td>1</td>
<td>83</td>
</tr>
<tr>
<td>2</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td>88</td>
</tr>
<tr>
<td>4</td>
<td>71</td>
</tr>
<tr>
<td>5</td>
<td>69</td>
</tr>
<tr>
<td>6</td>
<td>72</td>
</tr>
<tr>
<td>7</td>
<td>58</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>81</td>
</tr>
<tr>
<td>10</td>
<td>74</td>
</tr>
</tbody>
</table>

From Table. 4.1, 4.2, 4.3 workers have different processing time to tasks, different worker cost according to the different work experience.

The mALB problem concerns with the assignment of the proper worker to the proper station and assigning the suitable task to the suitable station in order to minimize the cycle time, minimize the variation of workload and minimize the total cost under the constraint of precedence relationships. In this study, the mALB problem subject to the following assumptions:

A1. The assembly line is capable of producing more than one type of product simultaneously; not in batches.

A2. The assembly of each model requires performing a set of tasks, which are connected by precedence relations.

A3. A subset of tasks is common to all models; the precedence graphs of all models can be combined into a non-cyclical joint precedence graph.

A4. Tasks, which are common to several models, are performed by the same station but they may have different processing times (i.e., zero processing times indicate that the task is not required for the model).

A5. The demands for all models during the planning period are fixed and known.

A6. A worker is assigned to one station, and will only process the tasks, which are assigned to that station.

A7. Worker processing time for each task is known and deterministic.

A8. Task processing time differs among workers because of their varying work experience and he/she can process all the tasks.

A9. Workers have different worker cost according to the different skill level. The worker cost is the same for each task in each model.

A10. A worker can process all the tasks for each model, and his/her work experience differs among tasks.
4.3 Mathematical Formulation

Considering the workers' capabilities in mALB problem, three objectives, minimization of the cycle time, minimization of the variation of workload and minimization of the total cost under the constraint of precedence relationship, are formulated. As described in section 3.3, these three objective are important in MS. Reducing cycle time means that can get higher producing efficiency in limited period. Decreasing variation of workload of machine can balance the usage of machine. Reducing the total worker cost means lower production cost.

The notation used in this Section can be summarized as follows.

**Notations**

**Indices**

- \(i\): index of the stations \((i = 1, \ldots, m)\).
- \(j, k\): index of the tasks \((j, k = 1, \ldots, n)\).
- \(w\): index of the workers \((w = 1, \ldots, m)\).
- \(l\): index of the models \((l = 1, \ldots, p)\).

**Decision variables**

\[
x_{jil} = \begin{cases} 
1, & \text{if task } j \text{ of model } l \text{ is assigned to station } i \\
0, & \text{otherwise}
\end{cases}
\]

\[
y_{iww} = \begin{cases} 
1, & \text{if worker } w \text{ is working in station } i \\
0, & \text{otherwise}
\end{cases}
\]

**Parameters**

- \(n\): number of tasks.
- \(m\): number of stations, number of workers.
- \(p\): number of models.
- \(d_{jw}\): worker cost of worker \(w\) process task \(j\).
- \(d_T\): total worker cost.
- \(r_l\): expected demand ratio of model \(l\) during the planning period.
- \(t_{jlw}\): processing time of task \(j\) for model \(l\) processed by worker \(w\).
- \(t_{jw}\): processing time of task \(j\) for combined model by worker \(w\).

\[
t_{jw} = \sum_{l=1}^{p} r_l t_{jlw} \tag{4.1}
\]

\[
c_T': \text{cycle time for combination of models.}
\]

\[
c_T: \text{cycle time for model } l.
\]

\[
c_T = \max_{1 \leq i \leq m} \left\{ \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jlw} x_{jil} y_{iww} \right\} \tag{4.2}
\]
Mathematical Formulation

\( \overline{c_T} \): average cycle time for models.

\[
\overline{c_T} = \frac{1}{p} \sum_{l=1}^{p} c_T^l
\]  

(4.3)

Suc\((j)\): the set of direct successors of task \( j \).
Pre\((j)\): the set of direct predecessors of task \( j \).
\( S_i \): the set of tasks assigned to station \( i \).
\( t_l(S_i) \): processing time at station \( i \) for model \( l \).

\[
t_l(S_i) = \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jlw} x_{jil} y_{iw}, \forall i
\]  

(4.4)

\( u_{il} \): utilization of the station \( i \) for model \( l \).

\[
u_{il} = t_l(S_i) / \overline{c_T}
\]  

(4.5)

\( \overline{u_l} \): average utilization of all stations for model \( l \).

\[
\overline{u_l} = \frac{1}{m} \sum_{i=1}^{m} u_{il}
\]  

(4.6)

Mathematical Model

The mathematical model for the mALB can be stated as follows:

\[
\text{minimize} \quad \{ c_T, v, d_T \}
\]

where

\[
c_T = \sum_{i=1}^{p} \left\{ r_l \max_{1 \leq i \leq m} \left( \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jlw} x_{jil} y_{iw} \right) \right\}
\]  

(4.7)

\[
v = \sum_{i=1}^{p} \left\{ r_l \sqrt{\frac{1}{m} \sum_{i=1}^{m} (u_{il} - \overline{u_l})^2} \right\}
\]  

(4.8)

\[
d_T = \sum_{i=1}^{m} \sum_{j \in S_i}^{m} \sum_{w=1}^{m} d_{jlw} y_{iw}
\]  

(4.9)

s.t.

\[
\sum_{i=1}^{m} i x_{jil} \geq \sum_{i=1}^{m} i x_{kil}, \forall k \in \text{Pre}(j), \forall j, l
\]  

(4.10)

\[
\sum_{i=1}^{m} x_{jil} = 1, \forall j, l
\]  

(4.11)

\[
\sum_{w=1}^{m} y_{iw} = 1, \forall i
\]  

(4.12)

\[
\sum_{i=1}^{m} y_{iw} = 1, \forall w
\]  

(4.13)

\[
x_{jil} \in \{0, 1\}, \forall j, i, l
\]  

(4.14)

\[
y_{iw} \in \{0, 1\}, \forall i, w
\]  

(4.15)

The first objective (Eq. 4.7) of the model is to minimize the cycle time of the assembly line based on demand ratio of each model. The second objective (Eq. 4.8) is to
minimize the variation of workload. The third objective (Eq. 4.9) is to minimize the
total worker cost. Inequity (4.10) states that all predecessor of task $j$ must be assign
to a station, which is in front of or the same as the station that task $j$ is assigned in.
Equation (Eq. 4.11) ensures that task $j$ must be assigned to only one station. Equation
(Eq. 4.12) ensures that only one worker can be allocated to station $i$. Equation (Eq.
4.13) ensures that worker $w$ can be allocated to only one station. Constrains (4.14) and
(4.15) represent the usual integrity restrictions.

There are three definitions for cycle time: Equation (Eq. 4.2) is cycle time of model
$l$, Equation (Eq. 4.3) is average cycle time for models and Equation (Eq. 4.7) is demand
ratio-based cycle time. What is different between these three cycle times. The Fig. 4.4
shows the comparison between three cycle times.

![Figure 4.4: Comparison of demand ratio-based cycle time and average cycle time for
the mALB](image)

If the assembly line just produce one model such as model 1 (model 2), the cycle
time of model 1 : 50 time units (model 2: 60 time units) can be used as a reference.
Average cycle time : 55 time units can be used when the assembly line processes two
models. However, using average cycle time as a reference when the demand ratio
of models is different can increase the idle time and decrease the line efficiency. The
demand ratio-based cycle time : 54 time units is calculated based on the demand ratio
of models (demand ratio of model 1 : 0.60, demand ratio of model 2 : 0.40). If the
demand ratio of model 1 is bigger than model 2, the cycle time will close to the cycle
time of model 1. On the contrary, the cycle time will toward cycle time of model 2. If
the demand ratio of models is same, the cycle time is the average time of two models.
Therefore, the demand ratio-based cycle time can reflect the relationship between the
demand ratio of models and cycle time.
4.4 Multiobjective Genetic Algorithm Approach

4.4.1 Main Ideas

As introduced in Section 3.4.1, genetic algorithms as one of evolutionary algorithms is used to resolve the optimization of multiple objectives simultaneously.

In this study, a new approach is proposed to solve the multiobjective mALB problem by simplifying the complexity of coding, reducing the computation time, and improving the quality of solutions with satisfying the minimization of cycle time, workload of station and worker cost concurrently for mALB problems considering workers’ capabilities.

As the same strategies as Section 3.4.1 for reducing the complexity of coding of this problem, a random key-based coding method considering the specific chromosome structure and the characteristics of the problem is adopted. Use this coding method can generate feasible solution easily to reduce searching solutions size.

Moreover, this real number based coding can easy apply arithmetic crossover processing to improve the abilities of exploration and exploitation of method.

Many moGAs differ mainly in the fitness assignment strategy, which is known as an important issue in solving moOPs. The awGA (as introduced in Section 2.2.1) utilizes some useful information from current population to readjust weights in order to obtain a search pressure towards to positive ideal point. It can improve the ability of convergence of solutions and increases the quality of solutions. However, the convergence speed of awGA is not so good because it doesn’t consider the dominance relationship of each individuals.

The proposed method uses a generalized Pareto-based scale-independent fitness function (gp-siff) considering the quantitative fitness values in Pareto space for both dominated and nondominated individuals. The gp-siff makes the best use of Pareto dominance relationship to evaluate individuals using a single measure of performance.

The gp-siff uses a pure fitness assignment strategy, which differs from the traditional Pareto-ranking methods. The gp-siff can assign discriminative fitness values not only to nondominated individuals but also to dominated ones. In traditional Pareto-ranking methods, the individuals with same rank value have same fitness value. For example, for each individual with rank value 1, the fitness value is same: 1. But, in gp-siff, the fitness value is different according to the number of dominated by this individual. So, the more the fitness value is, the better the individual will be.

The gp-siff mechanism can improve the ability of convergence of solutions more and increase the quality of solutions more than awGA.

Different from the selection strategy of Chapter 3, a binary tournament selection mechanism is used to generate new population. It choices two individuals randomly from the population and copy the better individual into the intermediate population. It can reduce the computation time more than RWS method because it doesn’t need complex calculation.

In this section, gp-siff approach using random key-based encoding method with multiple vectors is adopted. In this approach, a combination of the arithmetic crossover operator, weight mapped crossover, swap mutation operator, immigration operator and a binary tournament selection operator were used as genetic operations, and the fitness assignment strategy was used to determine fitness value.

The overall structure of adopted gp-siffGA is given as follows:
procedure: gp-siffGA for mALB problem
input: data set of mALB problem, GA parameters \((\text{popSize}, \text{maxGen}, p_M, p_C, p_I)\)
output: Pareto optimal solutions \(E\)

begin
\(t \leftarrow 0;\)
initialize \(P(t)\) by random key-based encoding routine;
calculate objectives \(c_T(P), v(P), d_T(P)\) by random key-based decoding routine;
calculate fitness \(\text{eval}(P)\) by gp-siff approach;
while (not terminating condition) do
create \(C(t)\) from \(P(t)\) by arithmetic and weight mapping crossover routine;
create \(C(t)\) from \(P(t)\) by swap mutation routine;
create \(C(t)\) from \(P(t)\) by immigration routine;
calculate objectives \(c_T(C), v(C), d_T(C)\) by random key-based decoding routine;
update Pareto \(E(P)\);
calculate fitness \(\text{eval}(P, C)\) by gp-siff routine;
select \(P(t + 1)\) from \(P(t)\) and \(C(t)\) by binary tournament selection routine;
\(t \leftarrow t + 1;\)
end
output Pareto optimal solutions \(E(P, C)\)
end

4.4.2 Genetic Representation

To optimize this problem under the constraint of precedence relationships at the same time, a random key-based chromosome representation method with multiple vectors, i.e. task priority vector (real number code), worker allocation vector (integral number code), task sequence vector (integral number code) and breakpoint vector (integral number code) is proposed.

The detailed encoding and decoding process of a chromosome consists of five phases:

Phase 1: Combining all the models

step 1.1: Combining all the models into a combined precedence graph.
step 1.2: Calculating the processing times for the combined model.

Phase 2: Creating a task sequence

step 2.1: Generating a random key-based chromosome by encoding method.
step 2.2: Creating a task sequence by decoding method.

Phase 3: Assigning worker to each station

step 3.1: Encoding for worker allocation by randomly assigning the worker to each station.

Phase 4: Assigning tasks to each station

Phase 5: Designing a schedule

step 5.1: Creating a schedule for the assembly line.
The chromosome includes 4 parts: task priority vector \( (v_1) \), worker allocation vector \( (v_2) \), task sequence vector \( (v_3) \), breakpoint vector \( (v_4) \). The task priority vector \( (v_1) \) used a real number to identify the priority level for one task generated by random key-base encoding method. The higher priority value of task is, the earlier the task start. the locus of each gene of worker allocation vector \( (v_2) \) represent station id and gene are presented as integer number, which is worker id assigned to this station. After applying random key decoding method, the task sequence vector \( (v_3) \) is generated. The breakpoint vector \( (v_4) \) is used to separate the task sequence into machine.

Phase 1: Combining all the models

step 1.1: Combining all the models into a combined precedence graph.

As an illustrative example, a mALB having 10 tasks, 4 stations, 4 workers and 2 models are used (see Fig. 4.2, 4.3, 4.4). For solving this mALB problem, the precedence relationship graph of two models need to be combined into one graph. The combined precedence relationship graph is shown in Fig. 4.5 and the processing time of combined model is shown in Table. 4.4

![Figure 4.5: Precedence relationship graph of combined model for the mALB](image)

Table 4.4: Data set of combined for mALB

<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessor</th>
<th>Task time unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( j )</td>
<td>( Pre(j) )</td>
<td>( w_1 )</td>
</tr>
<tr>
<td>1</td>
<td>Ø</td>
<td>17.4</td>
</tr>
<tr>
<td>2</td>
<td>Ø</td>
<td>8.4</td>
</tr>
<tr>
<td>3</td>
<td>Ø</td>
<td>18.0</td>
</tr>
<tr>
<td>4</td>
<td>{1}</td>
<td>11.6</td>
</tr>
<tr>
<td>5</td>
<td>{2,3}</td>
<td>25.0</td>
</tr>
<tr>
<td>6</td>
<td>{4}</td>
<td>16.8</td>
</tr>
<tr>
<td>7</td>
<td>Ø</td>
<td>25.2</td>
</tr>
<tr>
<td>8</td>
<td>{6}</td>
<td>21.0</td>
</tr>
<tr>
<td>9</td>
<td>Ø</td>
<td>19.0</td>
</tr>
<tr>
<td>10</td>
<td>{7,8,9}</td>
<td>30.0</td>
</tr>
</tbody>
</table>

step 1.2: Calculating the processing times for the combined model.

The expected demand ratio of two models during the planning period is set to 0.6 and 0.4, respectively. The calculated processing time by Eq. 4.4 for combined model is shown in Table 4.4.
Phase 2: Creating a task sequence

step 2.1: Generating a random key-based chromosome by encoding method.

The chromosome is composed of two parts (see Fig. 4.6); the first part is the task priority vector and the second part is worker allocation vector. A random method are used to generate the two vectors. The procedure of encoding for task priority vector and worker allocation vector are shown in Fig. 3.4 and Fig. 3.5.

Figure 4.6: Task priority and worker allocation vector of the example for mALB

step 2.2: Creating a task sequence by decoding method.

At the beginning, try to find a task for the first station. Task 1, 2, 3, 7, 9 are eligible for the position, which can be easily fixed according to adjacent relation among tasks. From the task priority vector, the priorities of them are 3.31, 7.14, 6.59, 4.13, and 10.72 respectively. The task 9 has the highest priority and is put into the task sequence. Then delete task 9 off the precedence relationship graph. Next, Task 1, 2, 3, 7 are eligible for the next position. Because node 2 has the largest priority value, it is put into the task sequence. The produce of this decoding a task sequence is shown in Fig. 3.6. The trace table for this procedure is shown in Table. 3.2. By the same manner, the chromosome can easily be encoded. The generated task sequence vector is shown in Fig. 4.7.

Figure 4.7: Task sequence vector of the example for mALB

Phase 3: Assigning worker to each station

step 3.1: Encoding for worker allocation by randomly assigning the worker to each station.

In this phase, the worker allocation vector \( v_2 \) is coded by randomly as shown in Fig. 3.3, which indicates the worker assigned for each station. The locus of worker allocation vector represents the station ID.

Phase 4: Assigning tasks to each station

The random key-based decoding procedure consists of three main steps:

The procedure of decoding for breakpoint vector
Step 1: Calculate the lower bound ($c_{LB}$) and the upper bound ($c_{UB}$) of the cycle time for the solution represented by the task sequence vector and worker allocation vector.

\[
c_{LB} = \left\lceil \frac{1}{m} \sum_{j=1}^{n} \min_{1 \leq w \leq m} \{t_{jw}\} \right\rceil \quad (4.16)
\]

\[
c_{UB} = \left\lceil \frac{1}{m} \sum_{j=1}^{n} \max_{1 \leq w \leq m} \{t_{jw}\} \right\rceil + \max_{1 \leq j \leq n} \{t_{jw}\} \quad (4.17)
\]

Step 2: Find out optimal cycle time by bisection searching.

Step 3: Partition the task sequence into $m$ parts with the optimal cycle time based on the worker allocation vector.

The procedure of decoding a breakpoint vector is shown in Fig. 3.8. The trace table for this procedure is shown in Table. 3.3. After applying the decoding procedure to the example, the breakpoint vector in Fig. 4.8 is obtained.

<table>
<thead>
<tr>
<th>Locus: gene id</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakpoint vector ($v_4$)</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 4.8: Breakpoint vector of the example for mALB

In this phase, this worker allocation vector ($v_2$) will decode into a solution with the task sequence vector ($v_3$), simultaneously by the breakpoint decoding procedure. It means that when the tasks are assigned into each station, the worker will be assigned into each station at the same time.

Based on the breakpoint vector, the task sequence vector is separated into 4 parts, all of which respectively correlated with one station. According to worker allocation vector, the workers are assigned to each station. Fig. 4.9 illustrates the phenotype of the example chromosome.

Phase 5: Designing a Schedule

Step 5.1: Creating a schedule for the assembly line.

Using the chromosomes, the schedule can be constructed as follows:

**Schedule of model 1** $S= (j, i, w, t_{jw})$

$S = \{(9, S_1, w_3, 0-17), (3, S_1, w_3, 17-44),$
\quad (5, S_2, w_4, 44-76), (7, S_2, w_4, 76-110),$
\quad (1, S_2, w_4, 110-127), (6, S_3, w_1, 127-155),$
\quad (8, S_4, w_2, 155-178), (10, S_4, w_2, 178-205)\}.$

**Schedule of model 2** $S= (j, i, w, t_{jw})$

$S = \{(9, S_1, w_3, 0-17), (2, S_1, w_3, 17-33), (3, S_1, w_3, 33-50),$
\quad (5, S_2, w_4, 50-72),$
\quad (1, S_2, w_4, 72-90), (4, S_3, w_1, 90-119),$
\quad (8, S_4, w_2, 119-152), (10, S_4, w_2, 152-179)\}.$
4.4.3 Genetic Operators

Considering the characteristic of random key-based genetic representation, arithmetic crossover operator (for worker allocation vector), weight mapped crossover (for task priority vector), swap mutation operator, immigration operator and a binary tournament selection operator are used.

Crossover operator

In this study, arithmetic crossover was used for task priority vector and weight mapped crossover (WMX) was used for worker allocation vector. Notation $p_{\text{c}}$ is defined as crossover probability.

**Arithmetic Crossover (for task priority vector)**

The details of arithmetic crossover are introduced at page 50.

**Weight Mapped Crossover (WMX) (for worker allocation vector)**

WMX generate the offspring by using segment of own parent to the left of the one-cut point, and then, remapping the right segment that base on the weight of other parent of right segment [4] (see page 50 for details).

Mutation operator

In this study, swap mutation for generating various offspring for task priority vector and worker allocation vector is used. Swap mutation selects two positions at random and then swaps the gene on these positions. Notation $p_{\text{m}}$ is defined as mutation probability.
Immigration operator

Please refer page. for details of immigration operator.

Selection operator

In this study, the selection operator of gp-siffGA uses a binary tournament selection, which works as follows. Choose two individuals randomly from the population and copy the better individual into the intermediate population.

4.4.4 Fitness Function

For the evaluation of individuals, generalized Pareto-based scale-independent fitness function (gp-siff) method were used.

The gp-siff can be as follows: Let the fitness value of an individual $S_i$ be a tournament-like score obtained from all participant individuals by the following function:

\[
eval(S_i) = p(S_i) - q(S_i), \quad i = 1, 2, ..., \text{popSize}
\]  

(4.18)

where

\[
p(S_i) = \{ |S_j| |f_k(S_j) - f_k(S_i) \geq 0, j = 1, 2, ..., \text{popSize}, j \neq i, k = 1, 2, 3 \}
\]  

(4.19)

\[
q(S_i) = \{ |S_j| |f_k(S_i) - f_k(S_j) \geq 0, j = 1, 2, ..., \text{popSize}, j \neq i, k = 1, 2, 3 \}
\]  

(4.20)

where $p(S_i)$ is the number of individuals, which can be dominated by the individual $S_i$, and $q(S_i)$ is the number of individuals which can dominate the individual $S_i$ in the objective space. An example is shown in fig. 4.10.

![Figure 4.10: An example of gp-siff](image-url)

The gp-siff uses a pure fitness assignment strategy, which differs from the traditional Pareto-ranking methods. The gp-siff can assign discriminative fitness values not
only to nondominated individuals but also to dominated ones. In traditional Pareto-ranking methods, the individuals with same rank value have same fitness value. For example, for each individual with rank value 1, the fitness value is same: 1. But, in gp-siff, the fitness value is different according to the number of dominated by this individual and the $q(S_i)$ is 0. So, the more the fitness value is, the better the individual will be.

4.5 Experiments and Discussion

As the same manner as Chapter 3, in order to illustrate the applicability of the proposed method, first try to solve the example problem in Section 4.2. A weight factor method was used to get a best compromise solution in Pareto set. The weight factors for cycle time objective, variation of workload objective, total worker cost objective were set to $w_1 = 0.5$, $w_2 = 0.2$ and $w_3 = 0.3$, respectively. The best compromise solution is shown in Fig. 4.11.

Figure 4.11: Gantt chart for the best compromised solution with $w_1=0.5$, $w_2=0.2$ and $w_3=0.3$ of mALB

In the real-world, the assembly line is not just for producing one unit of the product. It should produce several units. Fig. 4.12 show the Gantt charts for five units of
product (model sequence is 1,2,1,1,2).

Figure 4.12: Gantt chart of the best compromise solution (5 products) with $w_1=0.5$, $w_2=0.2$ and, $w_3=0.3$ of mALB

Gantt chart and schedule of the best compromised solution for weight $(w_1=1, w_2=0, w_3=0)$, $(w_1=0, w_2=1, w_3=0)$ and $(w_1=0, w_2=0, w_3=1)$ are shown in Fig. [A.5, A.6] and [A.7].

In order to evaluate the performance of the proposed method, a large set of problems are used. 3 representative precedence graphs from Scholl [90] are collected, which are widely used in the ALB literature. These precedence graphs contain with 35, 70 and 148 tasks. From each precedence graph, 4 different mALB problems are generated by using different WEST ratios: 3, 5, 7, 10 and 15. WEST ratio, as defined by Dar-EI [141], measures the average number of activities per station. The details of this benchmark are shown in Table 4.5. For each problem, the number of workstation is equal to the number of workers, and each task can be processed on any worker. The worker cost data are generated according to work skill level.

Table 4.5: Information for three benchmark problems for mALB

<table>
<thead>
<tr>
<th>Problem</th>
<th>No. of tasks</th>
<th>No. of stations</th>
<th>WEST ratio</th>
<th>No. of models</th>
<th>Demand ratio of each model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>4</td>
<td>8.75</td>
<td>3</td>
<td>0.4, 0.4, 0.2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>7</td>
<td>7.00</td>
<td>3</td>
<td>0.4, 0.4, 0.2</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>5</td>
<td>5.00</td>
<td>3</td>
<td>0.4, 0.4, 0.2</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2.92</td>
<td>3</td>
<td>0.4, 0.4, 0.2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>7</td>
<td>10.00</td>
<td>5</td>
<td>0.2, 0.2, 0.1, 0.1, 0.4</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>7</td>
<td>7.00</td>
<td>5</td>
<td>0.2, 0.2, 0.1, 0.1, 0.4</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>5</td>
<td>5.00</td>
<td>5</td>
<td>0.2, 0.2, 0.1, 0.1, 0.4</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>3.68</td>
<td>5</td>
<td>0.2, 0.2, 0.1, 0.1, 0.4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>148</td>
<td>10</td>
<td>14.80</td>
<td>7</td>
<td>0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td>10.57</td>
<td>7</td>
<td>0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>21</td>
<td>7</td>
<td>7.05</td>
<td>7</td>
<td>0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2</td>
</tr>
<tr>
<td>12</td>
<td>29</td>
<td>5.10</td>
<td>7</td>
<td>0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2</td>
<td></td>
</tr>
</tbody>
</table>

All the simulation experiments were performed on Pentium 4 processor (2.8 GHz
(clock), the program was written in C# language. The adopted parameters of the gp-siffGA are listed as following:

- Population size, \( \text{popSize}=200 \);
- Maximum generations, \( \text{maxGen}=500 \);
- Crossover probability, \( p_C=0.60 \);
- Mutation probability, \( p_M=0.20 \);
- Immigration probability, \( p_I=0.06 \).

Comparing the results of proposed GA with random key-based Genetic Algorithm (rkGA) approach (see Chapter 3) is used to examine its efficiency. Each approach is run 30 times. The rkGA used the same random key-based genetic representation and different genetic operators: arithmetic crossover operator (for worker allocation vector), weight mapped crossover (for task priority vector), swap mutation operator, Roulette Wheel Selection (RWS) operator. For evaluation function, the Adaptive Weight Approach (AWA) was used for rkGA.

**Coverage** \( C(A, B) \)

In order to evaluate the efficiency of the different approaches, the coverage method proposed by Zitzler was used. The coverage \( C(A, B) \) of two solution sets \( A \) and \( B \) is the percent of the individuals in \( B \) who are weakly dominated by individuals in \( A \). The larger \( C(A, B) \) is, the better performed by \( A \) than \( B \).

\[
C(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|} \tag{4.21}
\]

The value \( C(A, B) = 1 \) means that all individuals in \( B \) are weakly dominated by \( A \). On the contrary, \( C(A, B) = 0 \) denotes that none of individuals in \( B \) is weakly dominated by \( A \). Because the \( C \) measure considers the weakly dominance relationship between two sets \( A \) and \( B \), \( C(A, B) \) is not necessarily equal to \( 1-C(B, A) \).

Fig. 4.13 depicts the coverage of \( C(\text{gp-siffGA, rkGA}) \) and \( C(\text{rkGA, gp-siffGA}) \) from 30 runs. For each run, the solution set of two algorithms are compared using the coverage.

The first plot on the left of Fig. 4.13a describes the coverage of \( C(\text{gp-siffGA, rkGA}) \) for solving the small problem 35 tasks with 4 stations. From this plot, the lowest value of \( C(\text{gp-siffGA, rkGA}) \) is 0.26 and the highest value is 0.82, the mean value is 0.57. The first quarter of coverage data of \( C(\text{gp-siffGA, rkGA}) \) from 30 runs is distributed from 0.26 to 0.42, the half of data is located from 0.42 to 0.69 and the last quarter of data is located from 0.69 to 0.82.

On the contrary, the lowest value of \( C(\text{rkGA, gp-siffGA}) \) is 0.07 and the highest value is 0.71, the mean value is 0.40 as shown in the first plot on the left of Fig. 4.13b. The first quarter of coverage data of \( C(\text{rkGA, gp-siffGA}) \) from 30 runs is distributed from 0.07 to 0.30, the half of data is located from 0.30 to 0.54 and the last quarter of data is located from 0.54 to 0.71. Therefore, the performance of gp-siffGA is better than rkGA for solving the small size problem with 35 tasks, 4 stations.

As the complexity of problems increases, Fig. 4.13 shows that more than 60-90% of the nondominated solutions obtained by rkGA are dominated by the nondominated solutions obtained by gp-siffGA in solving the problems 35 tasks with 5, 7, and 12

---

76  Multiobjective mALB with Novel Cycle Time Reference
Experiments and Discussion

Figure 4.13: Box plots based on the coverage for mALB

(a) The coverage $C_{(gp-siffGA, rkGA)}$

(b) The coverage $C_{(rkGA, gp-siffGA)}$
stations, 70 tasks with 7, 10, 14, and 19 stations, 148 tasks with 10, 14, 21, and 29 stations.

From Table 4.6, it is clear that gp-siffGA could obtain obviously better efficacy (about 66% improved in convergence performance) performance.

Table 4.6: Comparison of the Coverage (mean and standard deviation) about gp-siffGA and rkGA for smALB

<table>
<thead>
<tr>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C(gp-siffGA, rkGA)</td>
<td>C(rkGA, gp-siffGA)</td>
</tr>
<tr>
<td>35(4) 0.5973</td>
<td>0.4563</td>
</tr>
<tr>
<td>35(5) 0.7087</td>
<td>0.2190</td>
</tr>
<tr>
<td>35(7) 0.7033</td>
<td>0.1523</td>
</tr>
<tr>
<td>35(12) 0.8150</td>
<td>0.01923</td>
</tr>
<tr>
<td>70(7) 0.7903</td>
<td>0.1017</td>
</tr>
<tr>
<td>70(10) 0.8260</td>
<td>0.0510</td>
</tr>
<tr>
<td>70(14) 0.8170</td>
<td>0.0510</td>
</tr>
<tr>
<td>70(19) 0.8437</td>
<td>0.0587</td>
</tr>
<tr>
<td>148(10) 0.8273</td>
<td>0.0683</td>
</tr>
<tr>
<td>148(14) 0.8163</td>
<td>0.0767</td>
</tr>
<tr>
<td>148(21) 0.8167</td>
<td>0.0860</td>
</tr>
<tr>
<td>148(29) 0.8163</td>
<td>0.0840</td>
</tr>
</tbody>
</table>

The Fig. 4.14 and Table 4.7 and shows the CPU times of each approaches with big parameters. It is easy to see that the gp-siffGA are faster than rkGA for small scale problems (35 tasks). For medium scale problems, the CPU time of two methods are almost same. As to see the big scale problem, the CPU time of three problems (10, 14, and 21 stations) applying proposed method are better than rkGA. Hence, for the 29 stations of 148 tasks, the CPU times of proposed is worse than rkGA. The reason of almost same CPU times for 70 tasks and worse result for one of 148 tasks is that the gp-siff need more calculation of each individuals for dominated and nondominated relationship. Generally speaking, better quality is more important than not bad CPU times.

The results indicate that the proposed approach could generate a set of optimal nondominated solutions effectively satisfying the multiple objectives concurrently for mALB problems considering workers’ capabilities with the help of demand ratio-based cycle time reference. Comparisons with existing moEA using same representation, crossover and mutation operator and different RWS operator, adaptive weight sum based fitness function, demonstrate that gp-siffGA could obtain both obviously better efficacy (about 66% improved in convergence performance) and efficiency (about 26% reduced in computation time) performances.
Figure 4.14: Comparison of means of CPU times measured by gp-siffGA and rkGA of mALB

Table 4.7: Comparison of the CPU time (mean and standard deviation) about gp-siffGA and rkGA for smALB
4.6 Summary

In this Chapter, for solving the multiobjective mALB problem considering workers’ capabilities, the mathematical models for minimization of cycle time, variation of workload of station, and total worker cost are extracted firstly.

For simplifying the complexity of coding, reducing the computation time, and improving the quality of solutions for multiobjective mALB problem considering different workers’ capabilities, a new multiobjective genetic algorithm is proposed.

The mALB can be simplified by composition and decomposition of models.

The demand ratio-based cycle time is proposed to reflect the relationship between the demand ratio of models and cycle time.

Generalized Pareto-based scale-independent fitness function (gp-siff) mechanism could improve the ability of convergence of solutions more and increase the quality of solutions more than rkGA.

The binary tournament selection mechanism could reduce the computation time.

Random key-based coding method could reduce the complexity of coding of this problem. In addition, applying genetic operators on this coding method could improve the abilities of exploration and exploitation.
Chapter 5

Fast and Effective Multiobjective Evolutionary Algorithm for PPS

This Chapter is organized as follows: section 5.1 introduces the background and related research works of problem. Section 5.2 describes the PPS problem description and formulates the mathematical model; Section 5.3 presents the detailed FE-MOEA approach; Section 5.4 gives the discussion and analysis of numerical experiments results; Finally, the conclusion and future work are given in Section 5.5.

5.1 Introduction

In a manufacturing system (MS), process planning and scheduling (PPS) is to process a set of prismatic parts into completed products effectively and economically. A prismatic part to be produced in the MS is generally described by features. Each feature has machining meanings associating with their geometric forms, such as holes, slots and bosses. For each feature, one or more corresponding operations are determined according to its feature geometry and available machining resources. Each operation includes the selection of machines, tools and tool access direction (TAD). There are precedence relationship constraints among operations according to the geometrical and technological consideration. Process planning is the determination of an optimal process plan, i.e. operations and their sequences, within the precedence relationship constraints and manufacturing resource (machine and tool) constraints. The scheduling is the allocation of the resources in the shop over time to manufacture the various parts. PPS becomes more important for the effective allocation and utilization of resources in the MS. However, seeking an optimal solution rapidly and effectively from all of the permutations, combinations of all of the tasks, manufacturing resources according to specified criteria is very difficult for decision maker.

Due to its importance, practicality and difficulty, in the past decade, many research works have addressed the PPS problem. These approaches include simulated annealing (SA) algorithm [1,54], tabu search algorithm [64], agent-based approach [66,67], particle swarm optimization (PSO) algorithm [34,68] and genetic algorithm (GA) [30,57,64].

Unfortunately, most of the above previous researchers concern single objective optimization rather than multiobjective optimization problems (moOPs). In the real world, almost every problem involves optimization of several incomparable or incom-
mensurable objectives simultaneously.

Multiobjective evolutionary algorithms (moEAs) have been recognized to be well-suited for solving moOPs. Several moEAs are proposed to solve moOPs directly, and present more compromising results than single objective optimization techniques.

As the first one of moEAs, vector evaluated genetic algorithm (VEGA) divides the population into $m$ sub-populations ($m$ is number of objectives), each of which evolves toward a single objective [69]. The benefit of VEGA is the strong ability to converge to the edge region of the Pareto front and less time complexity. However, the qualities (especially, diversity) of VEGA are not good because of the selection bias. Therefore, VEGA can not be used to address multiobjective PPS problems in practical situation.

As two classical moEAs, NSGA-II [71] and SPEA2 [72] have been proven to be able to get better quality in solving moOPs. NSGA-II can get the better quality owing to its Pareto ranking and crowding distance mechanism. SPEA2 depends on raw fitness assignment mechanism and density mechanism. Unfortunately, these two algorithms both need much CPU time. These drawbacks of NSGA-II and SPEA2 cause that they can not be applied to solve multiobjective PPS problems online.

Ho et al. propose a generalized Pareto-based scale-independent fitness function (GP-SIFF) to solve the large parameter optimization problem [94]. The GP-SIFF can obviously speed up the convergence rate, especially around the central area of the Pareto front. But the lack of diversity preservation mechanism causes that GP-SIFF also can not be chosen to deal with multiobjective PPS problems. Moreover, the difference between nondominated and dominated individuals can be decreased by GP-SIFF values. This disadvantage causes that more dominated individuals could be held in archive (external population) while nondominated ones would be removed from archive. It reduces the performance of archive, unless an enough large size of its archive is set to store a sufficient number of individuals.

To increase the quality both of convergence and dispersion, and to reduce the computation time, a new fast and effective multiobjective evolutionary algorithm (FE-MOEA) is proposed to solve the multiobjective PPS problem.

For covering the drawback of GP-SIFF, a new Pareto dominating and dominated relationship-based (PDDR) fitness function is proposed to evaluate the individuals. It gives the sensible difference values between the nondominated and dominated individuals. Moreover, ones with different numbers of dominating are also set as different fitness values even though they are all nondominated individuals. The individuals locating around the central region of Pareto font will have smaller values than the edge points.

The proposed method deliberately unites the two different mechanisms. One is the VEGA with a preference for the edge region of the Pareto front, and the other is the PDDR with tendency converging toward the central area of the Pareto front. These two mechanisms not only preserve the convergence rate, but also guarantee the better dispersion performance. Moreover, some problem-dependent crossover, mutation and local search methods are also used to improve the performance of the algorithm.
5.2 Multiobjective PPS Problem

5.2.1 Problem Description

The illustration of the PPS problem in a MS is shown in Fig. 5.1. In Fig. 5.1, there are a set of parts (four parts) to be processed by a number of machines (four machines) with a number of tools and different TADs. Each part has several operations (four parts have 4, 3, 3, and 4 operations respectively), and each operation can be performed on more than one suitable machines with different processing times. Table 5.1 shows the operation information of the 4 parts. Each column describes the part ID, operation ID, successors, operation name, TAD candidates, machine candidates, tool candidates and machining time, respectively.

The PPS problem herein is to determine a process plan and schedule which tells decision maker how, when and in which sequence to effectively allocate these operations of parts to suitable manufacturing resources. Moreover, this process plan and schedule should satisfy the multiply objectives concurrently while maintaining the feasibility of its. This problem can be defined as follows.

Operation sequencing: determine the operation sequences for all the parts so that the precedence relationships among all the operations are not violated.

Operation selection: determine the resources selection according to the feature geometry and available machining resources.

Part scheduling: determine how and when to allocate the manufacturing resources to the parts.

5.2.2 Mathematical Formulation

As the two objectives of PPS problems, minimizing makespan and minimizing variation of workload for each machine are important and frequently-used. Reducing makespan means that the MS can get higher producing efficiency in a limited period.
Decreasing variation of machine workload can balance the utilization of machines. The mathematical model of them is expressed in the following notations.

Indices

\( i, k \): indices of parts, \((i, k = 1, 2, \ldots, I)\).

\( j, l \): indices of operation for parts \( i \),
\((j, l = 1, 2, \ldots, J_{i})\).

\( m \): index of machines, \((m = 1, 2, \ldots, M)\).

\( t \): index of tools, \((t = 1, 2, \ldots, L)\).

\( d \): index of TADs, \((d = 1, 2, \ldots, D)\).

Decision variables

\[ x_{mij}^{M} = \begin{cases} 1, & \text{if } o_{ij} \text{ is performed by } m, \\ 0, & \text{otherwise} \end{cases} \]

\[ x_{lij}^{T} = \begin{cases} 1, & \text{if } o_{ij} \text{ is performed by } l, \\ 0, & \text{otherwise} \end{cases} \]

\[ x_{dij}^{D} = \begin{cases} 1, & \text{if } o_{ij} \text{ is performed by } d, \\ 0, & \text{otherwise} \end{cases} \]

\[ y_{ijklh} = \begin{cases} 1, & \text{if } o_{ij} \text{ is performed directly before } o_{klr}, \\ 0, & \text{otherwise} \end{cases} \]

Parameters

\( I \): number of parts.

\( J_{i} \): number of operations for part \( i \).

\( M \): number of machines.

\( L \): number of tools.

\( D \): number of TADs.

\( O_{i} \): set of operations for part \( i \).
Multiobjective PPS Problem

\[ O_i = \{ o_{ij} | j = 1, 2, \ldots, J_i \} \]

\( o_{ij} \): the \( j \)-th operation of part \( i \).

\( m_{ij} \): the \( m \)-th machine.

\( t_l \): the \( l \)-th tool.

\( a_d \): the \( d \)-th TAD.

\( M_{ij} \): set of machines that can process \( o_{ij} \).

\( A_{mj} \): set of operations that can be processed on machine \( m \).

\( r_{ijh} \): precedence constraints.

\( r_{ijh} = 1 \), if \( o_{ij} \) is predecessor of \( o_{ih} \); 0, otherwise.

\( t_{MC} \): machining time of \( o_{ij} \) by machine \( m \).

\( t_{MC} \): machine change time. It is needed when two adjacent operations belong to 1) different parts, or 2) same part and different machines.

\( t_{TC} \): tool change time. It is needed when two adjacent operations belong to 1) different parts, or 2) same part and different machines, or 3) same part, same machine and different tools.

\( t_{SC} \): set-up change time. It is needed when two adjacent operations belong to 1) different parts, or 2) same part and different machines, or 3) same part, same machine and different TADs.

\( t_{PRE} \): preparation time of operation \( o_{ij} \) by machine \( m \). The preparation time for an operation consists of machine change time, tool change time and set-up time for the operation.

\[ t_{PRE} = t_{MC} + t_{TC} + t_{SC} \quad (5.1) \]

\( t_{P} \): processing time of operation \( o_{ij} \) by machine \( m \). The processing time for an operation consists of the preparation time and the machining time for the operation.

\[ t_{P} = t_{PRE} + t_{M} \quad (5.2) \]

\( u_m \): workload of machine \( m \).

\[ u_m = \sum_{i=1}^{I} \sum_{j=1}^{J_i} t_{M}^{M_{ij}} x_{M_{ij}} \quad (5.3) \]

\( \bar{u} \): average workload of machine.

\[ \bar{u} = \frac{1}{M} \sum_{m=1}^{M} u_m \quad (5.4) \]

\( t_{C} \): completion time of \( o_{ij} \) by machine \( m \).

**Mathematical Model**

The mathematical model for minimization of makespan and variation of workload can be stated as follows:

\[
\text{minimize} \quad \{ t_{MC}, v^W \} \\
\text{where} \quad t_{MC} = \max_{m,ij} t_{M}^{M_{ij}} \\
v^W = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (u_m - \bar{u})^2} \quad (5.5)
\]
\begin{align*}
\text{s.t. } & (t^C_{mkh} - t^P_{mkh} - t^C_{mij}) x^M_{mij} x^M_{mkh} y_{ijkh} \geq 0, \\
& \forall (i, j), (k, h), \forall m \tag{5.7} \\
& r_{ijkh} y_{iij} = 0, \forall (i, j), h \tag{5.8} \\
& y_{ijij} = 0, \forall (i, j) \tag{5.9} \\
& \sum_{m=1}^{M} x^M_{mij} = 1, \forall (i, j) \tag{5.10} \\
& x^M_{mij} = 0, \forall (i, j) \notin A_m, \forall m \tag{5.11} \\
& y_{ijkh} \in \{0,1\}, \forall (i, j), (k, h) \tag{5.12} \\
& x^M_{mij} \in \{0,1\}, \forall m, (i, j) \tag{5.13} \\
& t^C_{mij} \geq 0, \forall m, (i, j) \tag{5.14}
\end{align*}

Equation \ref{eq:5.5} describes the one objective of minimization of makespan \(t^M\) and the other objective, minimization of variation of workload \(v^W\) is defined as Eq. \ref{eq:5.6}. From Eq. \ref{eq:5.1} and \ref{eq:5.2}, the main impact on Eq. \ref{eq:5.5} comes from Eq. \ref{eq:5.1} except the machining time Eq. \ref{eq:5.2}. The more changeovers (machine change time, tool change time, set-up change time) can increase makespan easily. On the contrary, less changeovers means that makespan can be reduced rather than more changeovers. However, minimizing changeovers means fewer changes between machines, tools and TADs. It causes that adjacent operations will be assigned to the same machine, or same tool, or same TAD with high probability rather than different machines, different tools and different TADs. As a result, the workload between machines will be unbalanced and variation of them will be increased correspondingly. Therefore, these two objectives are conflicting in PPS problem. Equation \ref{eq:5.7} imposes that any machine cannot be selected for one operation until the predecessor is completed. The precedence constraint is defined as Eq. \ref{eq:5.8} Equation \ref{eq:5.9} ensures the feasible operation sequence. The feasible resource selection are defined as Eq. \ref{eq:5.10} and \ref{eq:5.11}. Equations \ref{eq:5.12}, \ref{eq:5.13} and \ref{eq:5.14} impose nonnegative condition.

\section*{5.3 Fast and Effective Multiobjective Evolutionary Algorithm}

\subsection*{5.3.1 PDDR Fitness Function}

The fitness assignment strategy plays an important role in moEAs. The GP-SIFF makes the best use of Pareto dominance relationship to evaluate individuals. The GP-SIFF of an individual \(s_i\) is calculated by the following function:

\begin{equation}
\text{eval}(s_i) = p(s_i) - q(s_i), i = 1, 2, ..., \text{popSize} \tag{5.15}
\end{equation}

where \(p(s_i)\) is the number of individuals, which can be dominated by the individual \(s_i\), \(q(s_i)\) is the number of individuals which can dominate the individual \(s_i\). The bigger value is better.

In GP-SIFF, the fitness values of nondominated individuals mainly lie in the number of individuals which are dominated by this individual, \(p(s_i)\), and \(q(s_i)\) is 0. Moreover, the individuals locating around the central region of the Pareto front with bigger domination area can dominate individuals more than edge region. It causes that the higher GP-SIFF values can tend the solutions toward the central of the Pareto front.
However, the difference between nondominated and dominated individuals can be decreased by GP-SIFF values (as shown in Fig. 5.2a). In Fig. 5.2b, individual A has bigger fitness value than B and C, while B > C. In fact, nondominated individuals C should be better than B because B is a dominated individual. This unclear difference between nondominated and dominated individuals causes that more dominated individuals could be kept while nondominated ones would be removed in the evolving process. It can reduce the convergence performance of algorithm, unless an enough large size of its archive is set to store a sufficient number of individuals.

For covering the disadvantage of GP-SIFF, a new PDDR-based fitness function is proposed to evaluate the individuals. The PDDR of an individual $s_i$ is calculated by the following function:

$$
\text{eval}(s_i) = q(s_i) + \frac{1}{p(s_i) + 1}, \quad i = 1, 2, ..., \text{popSize}
$$

The smaller value is better.

According to the Eq. (5.16) PDDR can set the obvious difference values between the nondominated and dominated individuals. If the individual belongs to nondominated one, its fitness value will not exceed one. The fitness value of individual which is dominated by other will exceed one. Furthermore, ones with different numbers of dominating are also given the different fitness values even though they are all nondominated individuals. It is obvious that the nondominated individuals locating around the central region of Pareto front with bigger domination area will have smaller values (near to 0) than the edge points (near to 1).

From Fig. 5.2b, individual C is better than B while C is worse than B in Fig. 5.2a. Moreover, nondominated individual A has smaller value than C because the $p(A) > p(C)$.

To calculate the PDDR values of the $N$ individuals in population, firstly, the first individual $s_1$ needs to be compared with other $N - 1$ individuals to calculate the dominating and dominated relationship. Then, $p(s_1), q(s_1)$ are used to calculate the PDDR value. Considering $m$ objectives, it needs $m(N - 1)$ comparisons for $s_1$ (as shown in Fig. 5.3). For the second individual $s_2$, since the $s_2$ and $s_1$ has been calculated, it only
needs to be compared with \( s_3 \) to \( s_N \) totaled \( m(N - 2) \) comparisons. Following the same process, the last second one \( s_{N-1} \) only needs to be compare with the last one \( s_N \) with \( m \times 1 \) comparisons. So the total number of comparisons is \( m(N - 1) + m(N - 2) + \cdots + m \times 2 + m \times 1 = mN(N - 1)/2 \). The total time complexity of the PDDR is \( O(mN^2) \). Moreover, without calculating distance makes PDDR much faster than other approaches with diversity preservation mechanism.

Figure 5.3: The comparison process of PDDR

### 5.3.2 The Proposed Algorithm

The PDDR has the advantage with the tendency converging toward the center area of the Pareto front, but drawback to the edge region. It causes bad dispersion performance. Although VEGA can not achieve better dispersion performance, VEGA has the ability to approach for the edge area rather than the center area of the Pareto front. So it is natural, reasonable and possible to combine these two methods to improve the overall performance and reduce the computation time of the algorithm.

Figure 5.4 shows the description of FE-MOEA. The strong convergence capability of VEGA and PDDR ensures that the FE-MOEA has the ability to converge to the true Pareto front both in central and edge regions (left-top area and right-bottom in Fig. 5.4). The preferences for the edge area of the Pareto front in VEGA and the central area of the Pareto front in PDDR guarantee that the FE-MOEA distributes along the Pareto front evenly. Moreover, less computing time makes that FE-MOEA has higher efficiency than other approaches.

Since the multiobjective PPS problem belongs to the classical combinatorial optimization and NP-hard problems, higher efficacy and efficiency performances make that FE-MOEA seems well suited to solve such kind of problems. In addition, some problem-dependent crossover, mutation and local search methods are used to improve
the performance of the algorithm.

5.3.3 Main Framework of FE-MOEA

The evolving process of one generation of FE-MOEA is shown in Fig. 5.5. \( A(t) \) represents the archive at generation \( t \) and \( P(t) \) represents the population at generation \( t \). The solution procedure of one generation includes 4 phases.

- **Phase 1: Selection**
  For two objectives PPS problem, individuals are selected with replacement according to objective 1 into sub population 1 while ignoring objective 2 until the size of the sub population 1 (half of population size) is reached. In the same manner, individuals are selected for objective 2 into sub population 2 without considering objective 1 until sub population 2 is full.
Phase 2: Generation of mating pool
In this study, the sub populations and $A(t)$ are combined to form a mating pool. In the mating pool, sub-pop-1 stores the good individuals for one objective, and sub-population 2 holds the good individuals for the other objective. The archive saves the individuals with good PDDR values. Since there are two objectives in this PPS problem, the sizes of those two sub populations and archive are set as the same as half the population size. Therefore, in the mating pool, one-third of the individuals serve one objective, one-third the other objective, and the left one-third both the two objectives. The archive mechanism tries to cover the selection bias of VEGA. These three parts of the mating pool make the solutions converge to the Pareto front evenly.

Phase 3: Reproduction and local search
Some problem-dependent crossover and mutation operators are used to reproduce new chromosomes. Moreover, a local search mechanism is proposed to improve the quality of individuals after the reproduction process.

Phase 4: Archive maintenance
The individuals of $A(t)$ and $P(t+1)$ are combined to form a temporary archive $A'(t)$. Thereafter, the PDDR values of all individuals in $A'(t)$ are calculated and sorted in an ascending order. The smallest $|A(t)|$ individuals in $A'(t)$ are copied to form $A(t+1)$.

5.3.4 Genetic Coding
The chromosome consists of multiple vectors, i.e. operation sequence vector ($v_1$), machine vector ($v_2$), tool vector ($v_3$) and TAD vector ($v_4$). The detailed initialization process of a chromosome includes three steps.

Step 1: Deciding operation sequence
Operations are randomly selected from eligible operations to generate feasible operation sequences. After one operation being selected, it will be removed from the precedence graph and the eligible operations will be updated. In the same manner, the feasible operation sequence can be easily generated.

Step 2: Assigning resources
Machine and tool are selected randomly for each operation from available machine and tool candidates. TAD is also selected randomly from all TAD candidates for each operation. The generated chromosome is shown in Fig. 5.6.

Step 3: Generating schedule
After generating the chromosome, the schedule can be generated. When generating it, an operation can be started whenever its predecessor has been finished and the machine to process it is available.
5.3.5 Genetic Operators

Crossover operator

In this study, one-cut point order-based crossover is adopted. This crossover is mainly applied according to operation sequence vector \((v_1)\). The locus of machine vector \((v_2)\), tool vector \((v_3)\) and TAD vector \((v_4)\) are changed with same alleles of operation sequence vector \((v_1)\). Notation \(p_C\) is defined as crossover probability.

The algorithm is described as follows.

Step 1: According to the length of a parent, a cut point is generated randomly. Then, each parent is divided into two parts, a left side and a right side.

Step 2: The left side of offspring 1 comes from the left side of parent 1. The operator constructs the right side of offspring 1 according to the order of operations in parent 2. According to this order, the operator constructs the right side of offspring 1 with operations of parent 2, whose operation IDs are the same as operations of the right side in parent 1.

Step 3: Offspring 2 is created in the same manner.

Mutation operator

Three resource mutation operators proposed by Zhang et al. \[30\] are used. Notation \(p_M\) are defined as mutation probability.

**Machine mutation** changes the machine to perform an operation so as to reduce the machine change times. The algorithm is described as follows.

Step 1: A node ID is chosen randomly, and the corresponding machine \(M_{old}\) of this node ID is marked as the old machine.

Step 2: A new machine \(M_{new}\) is chosen randomly from the alternative machine set of this node ID to replace the old machine.

Step 3: Mark all the other machines in the machine vector \(v_2\) whose old machine is \(M_{old}\). If there exists an alternative machine \(M_{new}\) in any one of \(v_2\), \(M_{old}\) is replaced with \(M_{new}\).

**Tool mutation** uses the similar manner to operate on tool vector \(v_3\) after the machine mutation.

**TAD mutation** also uses the similar mechanism to operate on TAD vector \(v_4\) after both the machine mutation and the tool mutation.

Selection operator

In this study, a binary tournament selection (BTS) with replacement is used. Two individuals are chosen randomly from the population and the better (smaller) one is copied into the mating pool.

5.3.6 Local Search

The local search is proposed to reduce the changeovers (the machine change times, tool change times and TAD change times). It is applied for every chromosome after the reproduction. The pseudo code of local search is listed as follows.
procedure: local search for minimizing the changeovers
begin
for $i = 1$ to popSize do
  $l \leftarrow$ length of $v_1$ of chromosome $s_i$;
  for $j = 1$ to $(l - 1)$ do
    $p_{j+1} \leftarrow$ part ID of $v_1(j + 1)$;
    $p_j \leftarrow$ part ID of $v_1(j)$;
    if $(p_{j+1} = p_j)$ then
      $m_j \leftarrow$ machine ID of $v_2(j)$;
      $m_{j+1}^A \leftarrow$ alternative machine set of $v_2(j + 1)$;
      if $m_j \in m_{j+1}^A$ then
        $v_2(j + 1) \leftarrow v_2(j)$;
        $t_j \leftarrow$ tool ID of $v_3(j)$;
        $t_{j+1}^A \leftarrow$ alternative tool set of $v_3(j + 1)$;
      end if
      if $t_j \in t_{j+1}^A$ then
        $v_3(j + 1) \leftarrow v_3(j)$;
        $a_j \leftarrow$ TAD ID of $v_4(j)$;
        $a_{j+1}^A \leftarrow$ alternative TAD set of $v_4(j + 1)$;
      end if
      if $a_j \in a_{j+1}^A$ then
        $v_4(j + 1) \leftarrow v_4(j)$;
      end if
    end if
  end if
end for
if $s_i' < s_i$ then // $s_i'$ dominates $s_i$
  $s_i \leftarrow s_i'$;
else if not $s_i < s_i'$ then
  for $j = 1$ to popSize do
    if $s_j' < s_j$ then
      $s_j \leftarrow s_j'$;
      break;
    end if
  end for
end if
end for
end
5.4 Experiments and Discussion

Because it is very difficult to get real world problem data on PPS, 4 parts problem (as shown in Fig. 5.7) suggested by Li and McMahon [1] and Li et al. [64] are used.

The operation information for 4 part are shown in Table. A.3, A.4, A.5, A.6. The precedence relationship for 4 part are shown in Table. A.7, A.8, A.9, A.10. The precedence relationship graph is shown in Fig. A.8.

All the simulation are performed on Core 2 Quad processor (3.0 GHz clock), and the program is written in C# language. The adopted parameters are listed as follows:

- population size, popSize=100;
- maximum generation, maxGen=2000;
- archive size, archiveSize=50;
- crossover probability, pC=0.70;
- mutation probability, pM=0.60.

FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2 are run 30 times to compare the results with each other. It should be noted that the same selection, crossover, mutation, local search, and different fitness functions and archive mechanisms for four methods are used. Moreover, the parameters of all four methods are the same, except for the size of archive. The archive sizes of FE-MOEA and GP-SIFFGA are set to be half the population size, 50, while of NSGA-II and SPEA2 are set to be the same as the population size, 100. GP-SIFFGA uses the same framework as FE-MOEA, except the different fitness function.
5.4.1 Performance Measures

Let $S_j$ be a solution set for each method ($j=1, 2, 3, 4$). $PF^*$ is a known reference Pareto solutions. Because no researches have published their reference Pareto solutions for this problem, therefore, $PF^*$ in this study comes from combining all of the obtained Pareto set with 30 runs by 4 methods. In this study, the following four performance measures are considered.

Coverage $C(S_1, S_2)$ is the percent of the individuals in $S_2$ which are weakly dominated by $S_1$. The larger $C(S_1, S_2)$ means that $S_1$ outperforms $S_2$ in convergence.

Generational distance $GD(S_j)$ finds an average minimum distance of the solutions of $S_j$ from $PF^*$. The smaller $GD$ of $S_j$ means better $S_j$ in approaching $PF^*$.

Hypervolume $HV(S_j)$ is the size of the dominated space by $S_j$. Larger $HV$ means better convergence performance. The reference point for computing $HV$ comes from the maximum values of two objectives while combining all of the obtained Pareto set with 30 runs by 4 methods.

Spacing $SP(S_j)$ is the standard deviation of the closest distances of individuals by $S_j$. Smaller $SP$ means better dispersion performance.

The $C$, $GD$ and $HV$ are used to verify convergence performance while $SP$ is used to check the dispersion performance. The box-and-whisker plots are used to depict the comparison results. A box-and-whisker plot separates the data into four equal parts. Each of these parts contains one-fourth of the data.

5.4.2 Discussion of the Results

Figure 5.8 shows the comparison of the obtained Pareto set by using FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2 with combining 30 runs.

Figure 5.9 shows the 50% attainment surface by using FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2 with 30 runs. From Fig. 5.8 and 5.9 it is clear that FE-MOEA...
Experiments and Discussion

Figure 5.9: 50\% attainment surface by FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2 is better than GP-SIFFGA, NSGA-II and SPEA2 around both central and edge region of the Pareto front.

Figure 5.10 and Table 5.2 show the numerical comparison of the box-and-whisker plots, mean and standard deviation for $C$, $GD$, $HV$, $SP$ and the CPU times by FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2.

From Fig. 5.10a,b,c, it is easy to see that the FE-MOEA is slightly better than GP-SIFFGA (about 27\% improved), and is obviously better than NSGA-II and SPEA2 on $C$ measure (about 87-94\% improved). The $GD$ measure as shown in Fig. 5.10d also indicates that FE-MOEA can get smaller value than NSGA-II and SPEA2 (about 60\% and 70\% reduced) while slight worse than GP-SIFFGA (about 18\% worse). For $HV$ measure, as shown in Fig. 5.10e, FE-MOEA is better than GP-SIFFGA, NSGA-II and SPEA2 (about 2\%, 11\% and 10\%, respectively).

Table 5.2: Comparison of the coverage (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Improved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C(FE\text{-MOEA, GP-SIFFGA})$</td>
<td>$C(GP\text{-SIFFGA, FE-MOEA})$</td>
<td>27.47%</td>
</tr>
<tr>
<td></td>
<td>0.3643</td>
<td>0.2899</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C(FE\text{-MOEA, NSGA-II})$</td>
<td>$C(NSGA-II, FE-MOEA)$</td>
<td>93.73%</td>
</tr>
<tr>
<td></td>
<td>0.9110</td>
<td>0.0137</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$C(FE\text{-MOEA, SPEA2})$</td>
<td>$C(SPEA2, FE-MOEA)$</td>
<td>87.30%</td>
</tr>
<tr>
<td></td>
<td>0.8963</td>
<td>0.0243</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.9: 50\% attainment surface by FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2
Table 5.3: Comparison of the generational distance (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2

<table>
<thead>
<tr>
<th>Mean GD(FE-MOEA) - GD(GP-SIFFGA)</th>
<th>Mean GD(FE-MOEA) - GD(NSGA-II)</th>
<th>Mean GD(FE-MOEA) - GD(SPEA2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>45.1530 - 38.2873 = 17.87%</td>
<td>45.1530 - 12.1243 = -59.73%</td>
<td>45.1530 - 129.4717 = -70.43%</td>
</tr>
</tbody>
</table>

Table 5.4: Comparison of the hypervolume (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2

<table>
<thead>
<tr>
<th>Mean HV(FE-MOEA) - HV(GP-SIFFGA)</th>
<th>Mean HV(FE-MOEA) - HV(NSGA-II)</th>
<th>Mean HV(FE-MOEA) - HV(SPEA2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>36431 - 1.9060 = 344288.2250 = 5.82%</td>
<td>36431 - 16.4074 = 329174.1510 = 10.67%</td>
<td>36431 - 16.4074 = 330088.3943 = 10.37%</td>
</tr>
</tbody>
</table>

Table 5.5: Comparison of the spacing (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2

<table>
<thead>
<tr>
<th>Mean SP(FE-MOEA) - SP(GP-SIFFGA)</th>
<th>Mean SP(FE-MOEA) - SP(NSGA-II)</th>
<th>Mean SP(FE-MOEA) - SP(SPEA2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.7397 - 63.5703 = -51.64%</td>
<td>30.7397 - 32.5767 = -5.64%</td>
<td>30.7397 - 36.0613 = -14.76%</td>
</tr>
</tbody>
</table>
The dispersion performances, \( SP \), as shown in Fig. 5.10, indicates that FE-MOEA is obviously better than GP-SIFFGA (about 52% reduced), while better than NSGA-II and SPEA2 (about 5% and 15%). Without special mechanism to preserve the diversity evenly, FE-MOEA can also achieve satisfactory dispersion performance.

In general, the efficacy performance of FE-MOEA is better than the combination of gp-siff and VEGA (about 27% improved in convergence performance, about 52% improved in dispersion performance) while the efficiency performance is equivalent. Furthermore, the efficacy performance of FE-MOEA is also better than NSGA-II and SPEA2 (about 87-94% improved in convergence performance, 6-15% improved in dispersion performance), and the efficiency performance is obviously better than theirs (about 23-49% reduced in computation time).

### 5.4.3 Discussion of Time Complexity

In this study, the same genetic representation, initialization, selection, crossover, mutation, local search, as well as the same population sizes and maximum generations are set. The different archive sizes, calculation methods of fitness values and archive maintenance mechanism are adopted. Therefore, the differences of the time complex-
ity among FE-MOEA, NSGA-II and SPEA2 mainly rely on the distance and fitness calculation (GP-SIFFGA has the same time complexity as FE-MOEA). The distance calculation requires much CPU time because it needs to do the square and square root calculation. The comparisons of time complexity are shown in Table 5.7.

When updating the archive set, these three approaches have little difference, except that SPEA2 may need more computation cost in its truncation process.

From the comparisons and analysis as shown in Table 5.7, it is clear that FE-MOEA is much faster than NSGA-II and SPEA2. Furthermore, experimental comparison based on same simulation environment also indicates the same result (as shown in Fig. 5.10).

Such better convergence and dispersion performances should mainly attribute to VEGA’s preference for the edge region of the Pareto front and PDDR’s tendency converging toward the center area of the Pareto front. They preserve better performances both in efficacy and efficiency. Especially, these two mechanisms can also keep diversity evenly without special dispersion mechanisms like NSGA-II and SPEA2.

5.5 Summary

In this study, a fast and effective multiobjective evolutionary algorithm (FE-MOEA) approach is proposed to solve the multiobjective PPS problem while considering minimization of makespan and minimization of variation of machine workload.

The multiple vectors genetic representation is proposed to optimize the process planning and scheduling simultaneously.

For covering the disadvantage of GP-SIFF, a new Pareto dominating and dominated relationship-based (PDDR) fitness function is proposed to evaluate the individuals.

The proposed method tactfully unites the advantages of VEGA and PDDR to solve multiobjective PPS problem.

VEGA has a preference for the edge region of the Pareto front and the PDDR-based archive mechanism has the tendency converging toward the center area of the Pareto front.

The proposed method could preserve both the convergence and dispersion performances while reducing the computation time.

The binary tournament selection, problem-dependent order-based crossover, resource-based mutation, local search could improve the abilities of exploration and exploitation.
Table 5.6: Comparison of the CPU time (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU(FE-MOEA)</strong></td>
<td><strong>CPU(GP-SIFFGA)</strong></td>
<td>Reduced (CPU(FE-MOEA) - CPU(GP-SIFFGA))/ CPU(GP-SIFFGA)</td>
</tr>
<tr>
<td>1278757</td>
<td>124.9947</td>
<td>2.30%</td>
</tr>
<tr>
<td><strong>CPU(FE-MOEA)</strong></td>
<td><strong>CPU(NSGA-II)</strong></td>
<td>Reduced (CPU(FE-MOEA) - CPU(NSGA-II))/ CPU(NSGA-II)</td>
</tr>
<tr>
<td>1278757</td>
<td>166.2977</td>
<td>-23.10%</td>
</tr>
<tr>
<td><strong>CPU(FE-MOEA)</strong></td>
<td><strong>CPU(SPEA2)</strong></td>
<td>Reduced (CPU(FE-MOEA) - CPU(SPEA2))/ CPU(SPEA2)</td>
</tr>
<tr>
<td>1278757</td>
<td>260.1717</td>
<td>-48.89%</td>
</tr>
</tbody>
</table>

Summary 99
Table 5.7: Comparison of time complexity

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distance calculation</strong></td>
<td></td>
</tr>
<tr>
<td>FE-MOEA</td>
<td>Without special dispersion preservation mechanism, the time complexity of FE-MOEA is $O(0)$.</td>
</tr>
<tr>
<td>$O(0)$</td>
<td></td>
</tr>
<tr>
<td>NSGA-II</td>
<td>NSGA-II requires sorting the population according to each objective function value to obtain adjacent relationship first. The crowding distance can be calculated after normalizing each objective function. As a result, the time complexity of NSGA-II is $O(mN\log N)$ [71].</td>
</tr>
<tr>
<td>$O(mN\log N)$</td>
<td></td>
</tr>
<tr>
<td>SPEA2</td>
<td>SPEA2 needs to calculate all the distances of each individual to all the other individuals first. Then SPEA2 gets the $k$-th nearest distance for each individual after sorting its distances. So, the time complexity of SPEA2 is $O(mN^2\log N)$ [72].</td>
</tr>
<tr>
<td>$O(mN^2\log N)$</td>
<td></td>
</tr>
<tr>
<td><strong>Fitness calculation</strong></td>
<td></td>
</tr>
<tr>
<td>FE-MOEA</td>
<td>FE-MOEA requires $mN(N - 1)/2$ comparisons to calculate the PDDR value. So, the time complexity of FE-MOEA is $O(mN^2)$.</td>
</tr>
<tr>
<td>$O(mN^2)$</td>
<td></td>
</tr>
<tr>
<td>NSGA-II</td>
<td>NSGA-II needs $mN^2$ comparisons to acquire the domination relationship and counts the number of dominating and dominated for each individual in the phase of rank 1. Next, NSGA-II visits the all $N$ individuals and marks the ones whose number of dominated is 0 with rank 1. NSGA-II still needs $(N - 1)^2$ comparisons for the other individuals’ ranks. The overall time complexity of NSGA-II is $O(mN^2)$ [71].</td>
</tr>
<tr>
<td>$O(mN^2)$</td>
<td></td>
</tr>
<tr>
<td>SPEA2</td>
<td>SPEA2 also requires $mN^2$ comparisons to store the dominating and dominated relationship and its counts for each individual. The strength of each individual is thus dominating number. Thereafter, for each individual $i$, SPEA2 accumulates the strength of individuals which dominate individual $i$ in all the $N$ individuals to get the raw fitness value. The total time complexity of SPEA2 is also $O(mN^2)$ [72].</td>
</tr>
<tr>
<td>$O(mN^2)$</td>
<td></td>
</tr>
</tbody>
</table>

$m$ is the number of objectives.

$N$ is the sum of the population size and archive size.
Chapter 6

Fast Multiobjective Genetic Algorithm with Archive for PPS in DMS

This Chapter is organized as follows: Section 6.1 introduces the background and related research works of problem. Section 6.2 describes the problem and assumptions. The mathematical formulations for this problem are extracted in Section 6.3. Section 6.4 introduces the details of approach to resolve this problem and experiments and discussion are addressed in Section 6.5. Last Section 6.6 summaries this Chapter.

6.1 Introduction

In the single factory MSs, the PP is used to plan manufacturing resources and operations for a part to ensure the application of good manufacturing practice during its production processes. It includes determining suitable machines and tools, selecting set-up plans and sequencing machining operations of the part. The scheduling is to handle a group of parts with temporal constraints and competitive manufacturing resources, and decide how and when to assign the manufacturing resources to the parts. The scheduling system specifies the schedule of manufacturing resources on each operation of the parts according to the importance of parts, availability of resources and time constraints. In a complex manufacturing situation, it is ideal to integrate the planning and scheduling more closely to achieve a global optimum in manufacturing, and increase the flexibility and responsiveness of the systems [1].

The PPS problems in DMS are much more complicated than classical PPS problems because they involve not only the PPS problems in each factory, but also the problems in an upper level of how to allocate the parts to suitable factories. In fact, these issues are highly interrelated and should be considered simultaneously, since once a part is allocated to a factory and processed, it is usually unable or uneconomical to transfer this work-in-progress part to another factory for the remaining operations. Moreover, the production scheduling(s) in the factories have to depend on the parts allocated [62].

Different from the PPS in single factory, the PPS in DMS is to process a set of parts with operations on machines belonging to different factories by assigning parts into factory, selecting suitable manufacturing resources (machines, tools and TADs) and
sequencing the operations so as to determine a solution (schedule), in which the precedence constraints among operations can be satisfied and the corresponding objectives can be achieved. This solution tells a production facility what to make, when, and on which equipment, to process the products effectively.

As a multiobjective PPS problem in DMS, seeking a set of optimal solution rapidly and effectively from all of the permutations, combinations of all of the operations, manufacturing resources according to specified criteria is very difficult for decision maker.

The moEAs as described in Chapter 2 have been recognized to be well-suited for solving moOPs because their abilities to exploit and explore multiple solutions in parallel and to find a widespread set of non-dominated solutions in a single run.

In the past decade, a number of research works have appeared to address the integration of PPS. To facilitate the optimization process, some optimization approaches, such as the genetic algorithm (GA), simulated annealing (SA) algorithm, Tabu search algorithm, agent-based approach and particle swarm optimization (PSO) algorithm, have been developed in the last decade and significant improvements have been achieved.

However, most of the above previous researchers just concern with satisfying single objective. In real world, almost every problem involves optimization of several incomparable or incommensurable and often competing objectives simultaneously. In recent years, many researcher have changed their interest from single objective to multiobjective optimization problems (moOPs).

Moreover, there is a lack of papers, which have studied PPS problems in DMS. The previous research works are mostly concerned with single factory environment. Because of the characteristics of PPS in DMS, the processing of PPS in DMS will be more difficult than PPS in single manufacturing system.

As proposed at Chapter 4 the gp-siffGA can improve convergence speed and increase the quality of solutions. Although it can get a good speed, but the quality of solution is not as good as those well-known algorithm. One of the reasons is missing the elitism mechanism.

The classical moEA, such as, NSGA-II, SPEA2, can get better quality because the Pareto ranking mechanism or raw fitness assignment mechanism, and crowding distance or density mechanism and elitism (archive) mechanism. Unfortunately, the time complexity of the two famous approaches is much larger than gp-siffGA.

Therefore, the proposed approaches cannot get a good balance between efficacy (quality, mainly in convergence performance) and efficiency (speed) performances.

In this study, for reducing the computation time, and increasing the quality of obtained Pareto solutions, reducing the complexity of coding, so as to maintenance a good trade-off between efficacy and efficiency performances, a fast multiobjective genetic algorithm with archive (fmoGA-A) mechanism is proposed to solve the multiobjective PPS problem in DMS considering to combine different factories locating at different geographical area to produce various parts with different resource constraints.
6.2 Process Planning and Scheduling Problem in Distributed Environments

In DMSs, the PPS is to process a set of parts with operations on machines in different factories located at different geographical environments by selecting suitable manufacturing resources and sequence the operations so as to determine a schedule in which the precedence constraints among operations can be satisfied and the corresponding two objectives, minimization of the maximum total processing time and minimization of the maximum variation of workload of machine, can be achieved.

The PPS are both essential functional modules in product development and manufacturing. The major considerations in PP include: (1) generating machining operations based on the features of a part to meet desired functional specifications and achieve good manufacturability; (2) identifying machining resources applicable to the operations; and (3) determining the set-up plan and operation sequence according to some cost-effective criteria. Therefore, a process plan for a part can be represented by a series of machining operations, applicable resources for the operations, set-up plans, operation sequence, etc.

The scheduling system in DMS determines suitable production schedules of manufacturing equipment in each factory distributed in different environments for manufacturing a set of parts. The production schedules give the loading sequences of the part to the manufacturing equipment and the starting times of the individual manufacturing processed of the parts in each factory. So, the integrated PPS in DMS became more complex.

The description of PPS in DMS are introduced as follow; \( P \) parts need to be processed in \( F \) factories. For each factory \( f \), such as factory 1, they have \( M_f \) machines and \( L_f \) tools can be used. All of operations of one part have precedence relationships, which are not violated in manufacturing process. Each operation has machine candidates, tool candidates, tool access directions, and associated machining times.

If part 1 is assigned into factory 1, all of operations of part 1 need to be processed in factory 1. For one part, the PP should be optimized in PPS. For All parts, the PPS should be considered simultaneously for each factory.

The Fig. 6.1 shows the example of PPS problem in DMS for four parts to be process in two factories.

From Fig. 6.1 there are four parts that can be machined by 4, 3, 3 and 3 operations on four machines, respectively. For the different parts, there are precedence constraints among the operations to process them. The precedence relationships are not violated in manufacturing process. For example, part 1 and part 3 are assigned in factory 1, while part 2, and 4 are allocated in factory 2. The operation sequence of factory 1 (\( o_{33} \rightarrow o_{11} \rightarrow o_{12} \rightarrow o_{31} \rightarrow o_{32} \rightarrow o_{13} \rightarrow o_{14} \)), and factory 2 (\( o_{21} \rightarrow o_{42} \rightarrow o_{44} \rightarrow o_{41} \rightarrow o_{43} \rightarrow o_{22} \rightarrow o_{23} \)) can be sequenced firstly. According to the operation sequence, the manufacturing resources of each factory are specified (machines, tools and TADs), the schedule can be determined accordingly.

For each factory, the total processing time is considered. For all factories, the latest (maximum) total processing time in one factory is important because this time is the finish time of all production. In addition, the maximum variation of workload of machines is other objective in this study so as to balance the workload of machines in each factory.

In this study, the PPS problem in DMS subject to the following assumptions:
A1. Each machine can only handle one operation at each time.
A2. Each operation will be completed before another operation will be loaded. Note that, once a part is allocated to a factory, all of its operations will be processed in that factory.
A3. Parts are independent, and part preemption is not allowed.
A4. The sequence of the operations of each part complies with manufacturing constraints.
A5. All parts, machines and tools are available at time zero simultaneously.
A6. Each operation is performed on a single machine, and each machine can only execute an operation at a time.
A7. The time for a set-up is identical and independent of specific operations. The time for a machine change or a tool change follows the same assumption.
A8. Machines are continuously available for production.
A9. Each factory has same machines and tools, and different process skill. It means that the processing time and cost are different in each factory.

6.3 Mathematical Formulation

A typical distributed scheduling problem generally consists of a number of factories and a number of parts. Each factory has a different number of machines and can produce various product types with different efficiency. Each part has several numbers of operations, and each operation can be performed on more than one suitable machine.
(but not all) with different processing times. The problems herein are to determine how to allocate these parts to suitable factories and then determine the production scheduling and PP in each factory in order to minimize the total processing time and the variation of workload of machines simultaneously.

**Notations**

**Indices**

- \( i, k \): index of parts, \((i, k = 1, 2, \ldots, I)\).
- \( j, h \): index of operations for part \( i \), \((j, h = 1, 2, \ldots, J_i)\).
- \( f \): index of factories, \((f = 1, 2, \ldots, F)\).
- \( m \): index of machines in factory \( f \), \((m = 1, 2, \ldots, M_f)\).
- \( l \): index of tools in factory \( f \), \((l = 1, 2, \ldots, L_f)\).
- \( d \): index of tool access directions in factory \( f \), \((d = 1, 2, \ldots, D_f)\).

**Decision variables**

- \( x_f^{ij} = \begin{cases} 1, & \text{if operation } o_{ij} \text{ is performed in factory } f, \\ 0, & \text{otherwise} \end{cases} \)
- \( x_m^{Mij} = \begin{cases} 1, & \text{if operation } o_{ij} \text{ is performed by machine } m_m, \\ 0, & \text{otherwise} \end{cases} \)
- \( x_l^{Tij} = \begin{cases} 1, & \text{if operation } o_{ij} \text{ is performed by tool } t_l, \\ 0, & \text{otherwise} \end{cases} \)
- \( x_d^{Dij} = \begin{cases} 1, & \text{if operation } o_{ij} \text{ is performed by TAD } a_d, \\ 0, & \text{otherwise} \end{cases} \)
- \( y_{ijkh} = \begin{cases} 1, & \text{if operation } o_{ij} \text{ is performed directly before operation } o_{kh}, \\ 0, & \text{otherwise} \end{cases} \)
- \( \Omega (X, Y) = \begin{cases} 1, & \text{if } X \neq Y, \\ 0, & \text{if } X = Y \end{cases} \)

**Parameters**

- \( I \): number of parts.
- \( J_i \): number of operations for part \( i \).
- \( F \): number of factories.
- \( M_f \): number of machines in factory \( f \).
- \( L_f \): number of tools in factory \( f \).
- \( D_f \): number of tool access directions in factory \( f \).
- \( O_i \): set of operations for part \( i \), i.e., \( O_i = \{ o_{ij} | j = 1, 2, \ldots, J_i \} \).
- \( o_{ij} \): the \( j \)-th operation for part \( i \).
- \( m_m \): the \( m \)-th machine.
- \( t_l \): the \( l \)-th tool.
- \( a_d \): the \( d \)-th tool access direction.
- \( M_{ij}^{f} \): set of machines that can process operation \( j \) of part \( i \) \((o_{ij})\) in factory \( f \).
- \( A_{m}^{f} \): set of operations that can be processed on machine \( m \) in factory \( f \).
two adjacent operations (o

Fast Multiobjective Genetic Algorithm with Archive for PPS in DMS

different part, or 2) same part and different machine, or 3) same part, same machine

GM: machining time of operation j of part i by machine m in a general factory.

Mij: machining time of operation j of part i by machine m in factory f.

Mij = GM \times pf

FM: total machining time in factory f.

FM = \sum_{i=1}^{l} \sum_{j=1}^{l} \sum_{m \in \{m' | x_{ij}^m = 1 \}} M_{ijm}^f (6.1)

FM, machine change time index of factory f. It is same for each machine change.

MC: machine change time of operation oij by machine m in factory f. It is needed when two adjacent operations \((o_{kh}, o_{ij}) \in \{ (o_{k'kh'}, o_{ij'}) | (o_{k'kh'} = s_{w-1}) \land (o_{ij'} = s_w) \}\) belong to 1) different part, or 2) same part and different machines.

FM = FC \times \Omega \left( \{m | x_{ijm} = 1 \}, \{m | x_{ijm} = 1 \} \right) (6.2)

TC: total machine change time of factory f.

TC = \sum \sum (m,i,j) \in \{ (o_{ij} = s_{w}) \land (x_{ijm} = 1), w = 2, ... , W_f \} FM_{ijm} (6.3)

TC, tool change time index of factory f. It is same for each tool change.

TCm: tool change time of operation oij by machine m in factory f. It is needed when two adjacent operations \((o_{kh}, o_{ij}) \in \{ (o_{k'kh'}, o_{ij'}) | (o_{k'kh'} = s_{w-1}) \land (o_{ij'} = s_w) \}\) belong to 1) different part, or 2) same part and different machine, or 3) same part, same machine and different tool.

FM = FC \times \Omega \left( \{m | x_{ijm} = 1 \}, \{m | x_{ijm} = 1 \} \right) (6.4)
The operation.

The preparation time for an operation consists of the preparation time and the machining time for each operation.

\[ t^\text{PRE}_f = t^\text{MC} + t^\text{TC} + t^\text{SC} \] (6.8)

\( t^\text{PRE}_{f_{ij}} \): preparation time of operation \( j \) of part \( i \) by machine \( m \) in factory \( f \). The preparation time for an operation consists of the machine change time, the tool change time, and the set-up time for the operation.

\[ t^\text{P}_{f_{ij}} \]: processing time of operation \( j \) of part \( i \) by machine \( m \) in factory \( f \). The processing time for an operation consists of the preparation time and the machining time for the operation.

\[ u_{fm} \]: workload of machine \( m \) in factory \( f \).

\[ \bar{u}_f \]: average workload of machine in factory \( f \).

\[ t^\text{C}_{f_{ij}} \]: completion time of operation \( j \) of part \( i \) by machine \( m \) in factory \( f \).
Mathematical Model

The mathematical model for the PPS in DMS can be stated as follows:

\[
\begin{align*}
\text{minimize} & \quad \{t^P, v^W\} \\
\text{where} & \quad t^P = \max_{1 \leq f \leq F} \left( t_{j}^{TM} + t_{j}^{TMC} + t_{j}^{TTC} + t_{j}^{TSC} \right) \\
& \quad v^W = \max_{1 \leq f \leq F} \sqrt{\frac{1}{M_f} \sum_{m=1}^{M_f} (u_{f,m} - \bar{u}_f)^2} \\
\text{s.t.} & \quad \left( t_{f,m,k,h}^C - t_{f,m,k,h}^P - t_{f,m,k,h}^C \right) x_{f,i,j}^{M} x_{f,k,h}^{M} y_{i,j,k,h} \geq 0, \\
& \quad \forall (i,j), (k,h), \forall f, m \\
& \quad r_{i,j,h} y_{i,j,i} = 0, \forall (i,j), h \\
& \quad y_{i,j,i} = 0, \forall (i,j) \\
& \quad \sum_{m=1}^{M_f} x_{f,i,j}^{M} = 1, \forall (i,j), f \\
& \quad x_{f,i,j}^{M} = 0, \forall (i,j) \notin A_{f,m}, \forall m, f \\
& \quad y_{i,j,k} \in \{0,1\}, \forall (i,j), (k,h) \\
& \quad x_{f,i,j}^{M} \in \{0,1\}, \forall f, m, (i,j) \\
& \quad t_{f,m,k,h}^C \geq 0, \forall f, m, (i,j) \\
\end{align*}
\]

In this study, the objectives of this problem are minimization of the maximum total processing time \(t^P\) (Eq. 6.12), minimization of the maximum variation of workload of machine \(v^W\) (Eq. 6.13). Equation 6.14 imposes that for any resource (machine), if cannot be selected for one operation until the predecessor is completed and also idle time must be considered. Equation 6.15 ensures that the precedence constraints are not violated. Equation 6.16

6.4 Fast Multiobjective Genetic Algorithm with Archive Approach

6.4.1 Main Ideas

Duo to the PPS problem in MS belong to classical combinatorial optimization problem and NP-hard problem. In addition, multiobjective optimization make this problem more difficult and complex. The traditional methods have been proved not suitable to deal with large problems for their computational inefficiency and limitations. Hence, evolutionary algorithms seem to be especially suited to multiobjective optimization because of their capabilities of exploration and exploitation solutions in multiple Pareto-optimal solutions in a single simulation run. Genetic algorithms as one of evolutionary algorithms is adopted to resolve the optimization of two objectives simultaneously.

Many moGAs differ mainly in the fitness assignment strategy, which is known as an important issue in solving moOPs [4]. This study uses a generalized Pareto-based scale-independent fitness function (gp-siff) considering the quantitative fitness
values in Pareto space for both dominated and non-dominated individuals. The gp-siff makes the best use of Pareto dominance relationship to evaluate individuals using a single measure, not considering the rank and other complex computation. Therefore, gp-siff can significantly speed up the computing times. The details of gp-siff will be described in Section 6.4.4.

Although, gp-siff has ability to speed up searching time, however, the quality of obtained Pareto solutions are not as good as classical moEAs, such as NSGA-II, SPEA2. The latter approaches can get better convergence and dispersion, hence, the seeking time are so long. Such a long computing time is mainly due to the calculation of ranking and crowding distance (for NSGA-II) or raw fitness and density (for SPEA2). The elitism mechanism, which keep the “best” solutions from losing in evolving process, also can make solutions toward better convergence and dispersion.

Why not to combine these approaches to speed up the computation time while get better quality.

In this study, a fast moGA with archive (fmoGA-A) combining the archive mechanism and gp-siff which are general purpose problem-independent techniques was proposed to deal with PPS in DMS.

For maintaining a set of nondominated individuals found up-to-now, the elitism mechanism named archive is proposed. The elitism mechanism keeps the “best” solutions from losing in evolving process. The individuals in archive as the members of mating pool to be used to do crossover and mutation.

For reducing the computation time, this study uses a gp-siff considering the quantitative fitness values in Pareto space for both dominated and non-dominated individuals. The gp-siff does not consider the rank and other complex computation. Therefore, gp-siff can significantly speed up the computing times.

Therefore, for reducing the complexity of coding of this problem, a random key-based coding method is adopted. Use this coding method can generate feasible solution easily to reduce searching solutions size and improve the abilities of exploration and exploitation of method.

The evolving process in one generation of fmoGA-A could be illustrated by Fig. 6.2. $A(t)$ represents the archive at generation $t$ and $P(t)$ represents the population at generation $t$.

![Figure 6.2: Evolving process in one generation of fmoGA-A for PPS in DMS](image-url)
For updating the archive while the evolving process, the best $|A(t)|$ individuals from combination of $A(t)$ and $P(t)$ according to gp-siff value are selected to generate the new archive at generation $(t + 1)$.

In this section, a fmoGA-A approach using random key-based encoding method is proposed. In this approach, a combination of the weight mapped crossover (WMX) and the arithmetic crossover operator, swap mutation operator and a roulette wheel selection operator were used as genetic operations, and the fitness assignment strategy was used to determine fitness value.

### 6.4.2 Genetic Representation

For an example, 4 parts with several operations need to be processed in 2 factories as shown in Fig. 6.3. The information for 2 factories is shown in Fig. 6.4.

#### Table: Information for 2 factories of PPS in DMS

<table>
<thead>
<tr>
<th>Factory-ID</th>
<th>Processing capacity</th>
<th>Machine change time (time unit)</th>
<th>Tool change time (time unit)</th>
<th>Set-up change time (time unit)</th>
<th>Machine change cost (cost unit)</th>
<th>Tool change cost (cost unit)</th>
<th>Set-up change cost (cost unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P_f$</td>
<td>$l_{MC}, i_M$</td>
<td>$l_{TC}, i_T$</td>
<td>$l_{SC}, i_S$</td>
<td>$c_{MC}, c_{m}$</td>
<td>$c_{TC}, c_{t}$</td>
<td>$c_{SC}, c_{s}$</td>
</tr>
<tr>
<td>fact1</td>
<td>1.2</td>
<td>60</td>
<td>20</td>
<td>40</td>
<td>80</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>fact2</td>
<td>0.8</td>
<td>40</td>
<td>15</td>
<td>30</td>
<td>60</td>
<td>7</td>
<td>20</td>
</tr>
</tbody>
</table>

The operation information for 4 part are shown in Table. A.3 A.4 A.5 A.6.
The precedence relationship for 4 parts are shown in Table. A.7
The precedence relationship graph is shown in Fig. A.8

An assumption is supposed that each factory has different process skill, so the machining time of each operation of part will be different according to different factory which is part assigned.

The genetic representation including multiple vectors, i.e. part sequence vector (integral number code), breakpoint vector (integral number code), operation priority vector (real number code), operation sequence vector (integral number code), machine and tool vector (integral number code) and tool access direction vector (integral number code) is proposed.

The detailed encoding and decoding process of a chromosome consists of four phases:

Phase 1: Assigning part into factory
  step 1.1: Creating a part sequence vector by a random encoding method.
  step 1.2: Generating a breakpoint vector by a random encoding method.

Phase 2: Deciding operation sequence in each factory
  step 2.1: Generating a random key-based chromosome by encoding method.
  step 2.2: Creating an operation sequence by decoding method.

Phase 3: Assigning machines and tools for each part
Phase 4: Assigning axis for each part

The chromosome includes 6 vectors: part sequence vector ($v_1$), breakpoint vector ($v_2$), operation priority vector ($v_3$), operation sequence vector ($v_4$), machine and tool vector ($v_5$) and tool access direction vector ($v_6$).

Phase 1: Assigning part into factory

step 1.1: Creating a part sequence vector by a random encoding method.

A randomly generation method is used to create part sequence vector ($v_1$) as shown in Fig. 6.5

<table>
<thead>
<tr>
<th>Locus id :</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part sequence $v_1$:</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 6.5: Part sequence vector ($v_1$) of the example of PPS in DMS

step 1.2: Generating a breakpoint vector by a random encoding method.

Four parts will be assigned into 2 factories, at least one part need assigned into one factory. A random method is used to create breakpoint vector. The Fig. 6.6 shows this assignment. From Fig. 6.6 the part 1 will be signed in factory 1 and part 2, 3, 4 will be located in factory 2.
Phase 2: Deciding operation sequence in each factory

**step 2.1:** Generating a random key-based chromosome by encoding method.

The operation priority vector \( v_3 \) for 4 parts is shown in Fig. 6.7.

**step 2.2:** Creating an operation sequence by decoding method.

Creating an operation sequence procedure is proposed by Gen and Cheng [27]. The operation sequence obtained should be not violated the precedence relationship of operations for each part.

At the beginning, try to find an operation for the part 1 in factory 1. The operation priority vector of part 1 is shown in Fig. 6.8. The operation 1 of part 1 is the only one eligible for the first position, so \( o_{11} \) is selected for the first position of operation sequence in factory 1. Then delete operation 1 off the precedence graph of part 1.

Next, operation 2, 3, 5, 6, 18 are eligible for the next position of operation sequence in factory 1, which can be easily fixed according to adjacent relation among operations. From the operation priority vector, the priorities of them are 2.88, 44.55, 49.45, 27.88, and 18.72 respectively. The operation 5 (\( o_{15} \)) has the highest priority and is put into the operation sequence. Then delete operation 5 off the precedence graph. Next, operation 2, 3, 6, 7, 11 and 18 are eligible for the next position. Because node 3 (\( o_{13} \)) has the largest priority value, it is put into the operation sequence.

By the same manner, the part of chromosome in factory 1 can easily be encoded. When finishing the part of chromosome in factory 1, the part of chromosome with three parts in factory 2 can be decoded as shown in Fig. 6.9.

The generated operation sequence vector \( v_4 \) is shown in Fig. 6.10.
Phase 3: Assigning machines and tools for each part

Randomly select machines and tools from the selected factory that can be used for manufacturing the operation. The generated machine and tool vector \((v_5)\) is shown in Fig. 6.11.

Phase 4: Assigning axis for each part

Randomly select one amongst all possible TADs for the operation. The generated tool access direction vector \((v_6)\) is shown in Fig. 6.12.

6.4.3 Genetic Operators

Considering the characteristic of fmoGA-A, WMX (for part sequence vector) and arithmetic crossover operator (for operation priority vector), swap mutation operator (for part sequence vector and operation priority vector) are used.
Fast Multiobjective Genetic Algorithm with Archive for PPS in DMS

Figure 6.12: Tool access direction vector ($v_6$) for 4 parts of the example of PPS in DMS

Crossover operator

Two crossover operators are used: WMX and arithmetic crossover.

**WMX (for part sequence vector)**

The details of WMX for part sequence are introduced at page. The notation $p_{C1}$ is defined as weight mapped crossover probability.

**Arithmetic crossover (for operation priority vector)**

In this study, arithmetic crossover is used for operation priority vector. The notation $p_{C2}$ is defined as crossover probability (see page. for details).

Mutation operator

Notation $p_{M1}$, $p_{M2}$ are defined as mutation probability for part sequence vector and operation priority (see page. for details).

Selection operator

In this study, roulette wheel selection (RWS) is used to process selection operator (see page. for details).

6.4.4 Fitness Function

The gp-siff has been introduced at page. It evaluates the fitness value according to not only nondominated individuals but also dominated ones. In traditional Pareto-ranking methods, the individuals with same rank value have same fitness value. Hence, in gp-siff, the fitness value is different with the number of dominated by this individual because the $q(S_i)$ is 0. So, the more the fitness value is, the better the individual will be.

6.5 Experiments and Discussion

Most of papers on PPS problems are generally treating small-scale PPS problem in single factory for demonstrating their solving methods with effectiveness, such as Li and Mcmahon discussed three parts and ten parts problems.
However, detail descriptions were not listed in their papers. In addition, most of the researchers developed their own MS with enterprise and they never open the real world problem data considering commercial secrets. It is very difficult to get real world problem data on PPS in DMS.

Therefore, 4 parts problem suggested by Li and Mcmahon [1] and Li et al. [64] are used and extended to 2 factories problem for comparing with our method proposed and previous research works.

All the simulation experiments were performed on Pentium 4 processor (2.8 GHz clock), the program was written in C# language. The adopted parameters of the fmoGA-A are listed as following:

- Population size, \( \text{popSize}=200 \);
- Maximum generations, \( \text{maxGen}=500 \);
- Crossover probability, \( p_{C1}=0.20, p_{C2}=0.50 \);
- Mutation probability, \( p_{M1}=0.20, p_{M2}=0.30 \);

The same instance of the problem are used and do the calculation for 30 times. Then compare the results founded by proposed fmoGA-A with other approaches such as adaptive weight genetic algorithm (awGA) [27], fmoGA (without archive), non-dominated sorting genetic algorithm II (NSGA-II) [71] and strength Pareto evolutionary algorithm 2 (SPEA2) [72]. In the experiment, same representation and genetic operators, different selection method, Fitness Function and elitism mechanism are used.

Let \( S_j \) be a solution set for each solution method \( (j=1, 2, \ldots, 5) \). \( PF^* \) is a known set of the Pareto-optimal set. In this study, the following three performance measures are considered.

**Coverage** \( C(S_1, S_2) \) is the percent of the individuals in \( S_2 \) who are weakly dominated by \( S_1 \). The value \( C(S_1, S_2) = 1 \) means that all individuals in \( S_2 \) are weakly dominated by \( S_1 \). On the contrary, \( C(S_1, S_2) = 0 \) denotes that none of individuals in \( S_2 \) is weakly dominated by \( S_1 \) [96]. The larger \( C(S_1; S_2) \) is, the better \( S_1 \) outperforms \( S_2 \) in \( C \).

**Generational distance** \( GD(S_j) \) finds an average minimum distance of the solutions of \( S_j \) from \( PF^* \). This measure calculates the closeness of a solution set \( S_j \) from the set \( PF^* \) [97]. The smaller \( GD \) for one solution set is, the better it is in approaching \( PF^* \).

**Hypervolume** \( HV(S_j) \) is the size of the dominated space by \( S_j \) [74]. The larger \( HV \) of a solution set is, the better it is.

Figure 6.13 shows the comparison of obtained Pareto set by using fmoGA-A, awGA and gp-fmoGA (without archive) with one run.

The Fig. 6.14 shows the comparison of generational distance, hypervolume and CPU times measures by fmoGA-A, awGA and fmoGA.

From Fig. 6.13 and 6.14 it is clearly that the fmoGA-A get better result than awGA and fmoGA, however, the computation times are bigger than the latter. This difference of computation times are negligible with comparing with the NSGA-II and SPEA2.

Figure 6.15 shows the comparison of obtained Pareto set by using fmoGA-A, NSGA-II and SPEA2 with one run.

From Fig. 6.15 it is not clearly which one is better by three approaches.
Figure 6.13: Comparison of obtained Pareto set by fmoGA-A, awGA and fmoGA of PPS in DMS

Figure 6.14: Comparison of generational distance, hypervolume and CPU times measures by fmoGA-A, awGA and fmoGA of PPS in DMS

For comparing the performance of proposed method with NSGA-II and SPEA2, generational distance and hypervolume are used. The Fig. 6.16 and 6.17 shows the comparison of box-and-whisker plots and means of generational distance and hypervolume by fmoGA-A, NSGA-II and SPEA2.

Table 6.16 shows the numerical comparison of mean and standard deviation for GD, HV and the CPU times by fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2.

From Fig. 6.16a and 6.17a, it is clearly that the NSGA-II > fmoGA-A > SPEA2 on generational distance performance measure. For hypervolume performance measure, as shown in Fig. 6.16b and 6.17b, SPEA2 > fmoGA-A > NSGA-II. From Fig. 6.16c and 6.17c, it is easy to see that the fmoGA-A are faster than NSGA-II and SPEA2.

The descriptive statistics such as mean of generational distance and hypervolume is not strong enough to differentiate the result of proposed approach from the NSGA-II and SPEA2. So, the hypothesis test is used to verify whether there exists statistical significance for the results.

The table 6.4 shows the result of the t test which is the most often used hypothesis test method in evolutionary algorithm community (μ^{GD}_{fmoGA-A}, the mean of generational distance by fmoGA-A, μ^{HV}_{fmoGA-A}, the mean of hypervolume by fmoGA-A).
Figure 6.15: Comparison of obtained Pareto set by fmoGA-A, NSGA-II and SPEA2 of PPS in DMS

Table 6.1: Comparison of the generational distance (mean and standard deviation) about fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GD(fmoEA-A)</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(awGA))/GD(awGA)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(NSGA-II))/GD(NSGA-II)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(SPEA2))/GD(SPEA2)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(fmoGA))/GD(fmoGA)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(NSGA-II))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(SPEA2))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(awGA))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(nsGA-II))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(SPEA2))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(awGA))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(nsGA-II))/GD(fmoEA-A)</td>
</tr>
<tr>
<td>Reduced</td>
<td>23.7877</td>
<td>(GD(fmoEA-A) - GD(SPEA2))/GD(fmoEA-A)</td>
</tr>
</tbody>
</table>
Table 6.2: Comparison of the hypervolume (mean and standard deviation) about fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hybridized</td>
<td></td>
</tr>
<tr>
<td>HV(fmoEA-A)</td>
<td>HV(awGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(HV(fmoEA-A) - HV(awGA)) / HV(awGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>164369.41</td>
<td>10 153510.61</td>
</tr>
<tr>
<td></td>
<td>60.70%</td>
<td>5940.0456</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3705.6633</td>
</tr>
<tr>
<td></td>
<td>fmoEA-A</td>
<td>fmoGA</td>
</tr>
<tr>
<td></td>
<td>Improved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(HV(fmoEA-A) - HV(fmoGA)) / HV(fmoGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>164369.41</td>
<td>148853.9620</td>
</tr>
<tr>
<td></td>
<td>10.42%</td>
<td>5940.0456</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5860.3460</td>
</tr>
<tr>
<td></td>
<td>NSGA-II</td>
<td>SPEA2</td>
</tr>
<tr>
<td></td>
<td>Improved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(HV(fmoEA-A) - HV(NSGA-II)) / HV(NSGA-II)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>164369.41</td>
<td>163815.1367</td>
</tr>
<tr>
<td></td>
<td>0.34%</td>
<td>5940.0456</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7098.8333</td>
</tr>
<tr>
<td></td>
<td>SPEA2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(HV(fmoEA-A) - HV(SPEA2)) / HV(SPEA2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>164369.41</td>
<td>185709.7480</td>
</tr>
<tr>
<td></td>
<td>-1.49%</td>
<td>5940.0456</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5732.1955</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison of the CPU time (mean and standard deviation) about fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hybridized</td>
<td></td>
</tr>
<tr>
<td>CPU(fmoEA-A)</td>
<td>CPU(awGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reduced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(CPU(fmoEA-A) - CPU(awGA)) / CPU(awGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>152.7100</td>
<td>45.4610</td>
</tr>
<tr>
<td></td>
<td>42.55%</td>
<td>5.2389</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.6891</td>
</tr>
<tr>
<td></td>
<td>CPU(fmoEA-A)</td>
<td>CPU(fmoGA)</td>
</tr>
<tr>
<td></td>
<td>Reduced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(CPU(fmoEA-A) - CPU(fmoGA)) / CPU(fmoGA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>152.7100</td>
<td>121.1023</td>
</tr>
<tr>
<td></td>
<td>177.62%</td>
<td>5.2389</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.8943</td>
</tr>
<tr>
<td></td>
<td>CPU(fmoEA-A)</td>
<td>CPU(NSGA-II)</td>
</tr>
<tr>
<td></td>
<td>Reduced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(CPU(fmoEA-A) - CPU(NSGA-II)) / CPU(NSGA-II)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>152.7100</td>
<td>917.8633</td>
</tr>
<tr>
<td></td>
<td>90.46%</td>
<td>5.2389</td>
</tr>
<tr>
<td></td>
<td></td>
<td>55.0972</td>
</tr>
<tr>
<td></td>
<td>CPU(fmoEA-A)</td>
<td>CPU(SPEA2)</td>
</tr>
<tr>
<td></td>
<td>Reduced</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(CPU(fmoEA-A) - CPU(SPEA2)) / CPU(SPEA2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>152.7100</td>
<td>1361.7323</td>
</tr>
<tr>
<td></td>
<td>-55.52%</td>
<td>5.2389</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.8242</td>
</tr>
</tbody>
</table>
Figure 6.16: Comparison of generational distance, hypervolume and CPU times measures by fmoGA-A, NSGA-II, and SPEA2 of PPS in DMS

Figure 6.17: Comparison of mean of generational distance, hypervolume and CPU times measures by fmoGA-A, NSGA-II, and SPEA2 of PPS in DMS

The $t$ test result is 1 indicates a rejection of the null hypothesis at the 5% significance level. 0 indicates a failure to reject the null hypothesis at the 5% significance level. From Table 6.4, it is clearly that the means of generational distance by fmoGA-A, NSGA-II and SPEA2 have statistical difference with significance level 5%, the means of hypervolume by fmoGA-A and SPEA2, NSGA-II and SPEA2 have statistical difference with significance level 5%. However, the means of hypervolume by fmoGA-A and NSGA-II do not have statistical difference with significance level 5%.

Because there exists no unary quality measure that is able to indicate whether an approximation $A$ is better than an approximation $B$ \[^{[100]}\], so, the efficacy performance of fmoGA-A is not less than NSGA-II and SPEA2.

Based on the above analysis, the solution produced by fmoGA-A is better than awGA and fmoGA did on efficacy performance and negligible difference on efficiency performance. Comparing to NSGA-II and SPEA2, the efficacy (convergence) performance of fmoGA-A is not less than NSGA-II and SPEA2 did and the efficiency performance is obviously better than theirs (about 83-89% reduced in computation time).

### 6.6 Summary

In this Chapter, for solving the multiobjective PPS problem in DMS, the mathematical models for minimization of the maximum total processing time and minimization of maximum variation of workload of machine are extracted.
For improving the efficacy performance and speeding up the efficiency when solving the PPS problem in DMS with satisfying the two objectives concurrently, fast multiobjective genetic algorithm with archive (fmoGA-A) approach is proposed. The multiple vectors genetic representation is proposed to optimize the process planning and scheduling simultaneously.

A generalized Pareto-based scale-independent fitness function (gp-siff) is adopted to evaluate the fitness function. It could significantly speed up the computing times. The archive mechanism could cover the selection bias to make the solutions converge the Pareto front. It could increase the quality of solutions. The replacement of archive according to gp-siff value also could enhance the quality of solutions. Random key-based coding method could reduce the complexity of coding of this problem. In addition, applying genetic operators on this coding method could improve the abilities of exploration and exploitation.

Table 6.4: t test result of fmoGA-A, NSGA-II and SPEA2 of PPS in DMS

<table>
<thead>
<tr>
<th>No.</th>
<th>Null hypothesis $H_0$</th>
<th>Alternative hypothesis $H_1$</th>
<th>$t$ test result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\mu_{fmoGA} = \mu_{NSGA}$</td>
<td>$\mu_{fmoGA} \neq \mu_{NSGA}$</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>$\mu_{fmoGA} &lt; \mu_{NSGA}$</td>
<td>$\mu_{fmoGA} \neq \mu_{NSGA}$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$\mu_{fmoGA} = \mu_{SPEA2}$</td>
<td>$\mu_{fmoGA} \neq \mu_{SPEA2}$</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$\mu_{NSGA} = \mu_{SPEA2}$</td>
<td>$\mu_{NSGA} \neq \mu_{SPEA2}$</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>$\mu_{fmoGA} = \mu_{SPEA2}$</td>
<td>$\mu_{fmoGA} \neq \mu_{SPEA2}$</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$\mu_{NSGA} = \mu_{SPEA2}$</td>
<td>$\mu_{NSGA} \neq \mu_{SPEA2}$</td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusions

The primary aim of this study is to present an investigation on multiobjective evolutionary algorithms (moEAs) for two real-world scheduling problems in machining and assembly processes: assembly line balancing (ALB) and process planning and scheduling (PPS) problems, which are important and intractable combinatorial optimization problems in MS.

For solving the above problems, this study developed approaches with the help of intelligent techniques, such as, evolutionary algorithms (EAs). EAs seem to be especially suited to multiobjective optimization problems because they are able to explore multiple Pareto-optimal solutions in a single simulation run and may exploit similarities of solutions by recombination.

For multiobjective smALB and mALB problems considering different workers’ capabilities, the mathematical models are suggested firstly. Next, two moEA approaches are proposed to deal with these problems. Through the experiments results, the proposed approaches could generate effectively a set of optimal Pareto solutions for these two problems with better quality and better speed.

For multiobjective PPS problem and PPS problem in DMS, the mathematical models are suggested firstly. Next, two moEA approaches named FE-MOEA and fmoGA-A are proposed to deal with these problems. Through the experiments results, the proposed approaches could generate effectively a set of optimal Pareto solutions for these two problems with better efficacy performance and better efficiency performance.

It is necessary to mention that the research order is smALB, mALB, PPS in DMS and PPS problems. The corresponding proposed method is rkGA, gp-siffGA, fmoGA-A and FE-MOEA respectively.

7.1 Multiobjective smALB Problem Considering Workers’ Capabilities

In smALB problem with different workers’ capabilities, the different workers’ capabilities cause that processing time and cost fall in conflict. Higher capability means shorter processing time and higher cost, and vice versa. Allocation of the tasks among stations satisfying the precedence relationship constraint between tasks and assignment proper worker to suitable station, so as to effectively generate a set of optimal nondominated solutions satisfying multiple objectives simultaneously is with considerable difficulties.
Therefore, issue 1 is optimization of multiobjective smALB problem efficiently with minimization objectives for cycle time, variation of workload of station, and worker cost concurrently considering various workers’ capabilities.

For optimization of smALB considering workers’ capabilities, a new multiobjective random key-based genetic algorithm (rkGA) is proposed.

A problem-dependent multiple vector coding is designed firstly. One individual is made up of task priority, task sequence, worker allocation and breakpoint vectors. Task priority and worker allocation are used to apply on crossover and mutation. Random key-based coding method could reduce the complexity of coding of this problem. The real number of task priority can be easy applied on arithmetic crossover to improve exploration capability.

Weight mapped crossover, swap mutation as well as immigration operators are used to enhance the searching performance. Roulette wheel selection (RWS) is adopted to generate next population. An adaptive weight sum based fitness function is applied to improve the convergence by normalizing multiple objective functions and transforming them into one value.

The performance of proposed method is validated through numerical experiments for solving the 35 tasks, 6-15 stations and 6-15 workers ALB problem. The results indicate that with considering workers’ capabilities, the proposed approach could generate a set of optimal nondominated solutions effectively satisfying the minimization of cycle time, workload of station and worker cost concurrently for smALB problems.

Comparing rkGA to priority key-based genetic algorithm (priGA) (different integer priority key-based coding, one cut point crossover from rkGA) indicates that rkGA could achieve both better efficacy (about 5% improved in convergence performance) and efficiency (about 48% reduced in computation time) performances than priGA.

### 7.2 Multiobjective mALB Problem with Novel Cycle Time Reference

As an extension of smALB, the mixed-model ALB problem produces different models of the same product on an assembly line. Multiobjective mixed-model ALB problem deals with the allocation of the tasks of different models among stations and assignment of the workers among stations to generate a set of optimal solutions satisfying multiple objectives simultaneously. Excepting the same consideration of workers’ capabilities, additional attentions about calculation of cycle time of multiple models need to be concerned.

Therefore, issue 1 is extended to more complex issue 2, which is optimization of multiobjective mALB problem effectively with same objectives as issue 1 considering workers’ capabilities. Different from issue 1, calculation of cycle time with multiple models becomes more difficult than single model.

In mALB, composition and decomposition method is adopted to simplify the mALB to smALB so as to the same coding method as Chapter 3 can be applied easily. Under consideration of workers’ capabilities for mALB, all of models are composed into one combined model firstly. Then, same coding method as smALB is applied to assign tasks of the combined model and workers to stations. Next, each task of each model can be decomposed into to each station accordingly. Different from traditional reference cycle time, a novel reference cycle time calculation method according to different
demand ratio of models is proposed. It can reflect the relationship between the demand ratio of models and cycle time. The mALB can be simplified by composition and decomposition of models.

To solve mALB, a generalized Pareto-based scale-independent fitness function (gp-siff) based genetic algorithm (gp-siffGA) is proposed. In traditional Pareto-ranking methods, the nondominated individuals with same rank value have same fitness value. However, the fitness value of gp-siff is different according to the number of dominated by this individual. The more the fitness value is, the better the individual will be. Therefore, gp-siff can enhance the ability of convergence of solutions more and increase the quality of solutions more than rkGA.

Same reproduction strategies as Chapter 3 are adopted and a binary tournament selection (BTS) is used instead of roulette wheel selection.

The binary tournament selection mechanism could reduce the computation time. Random key-based coding method could reduce the complexity of coding of this problem. In addition, applying genetic operators on this coding method could improve the abilities of exploration and exploitation.

The performance of proposed method is validated through numerical experiments for solving the mALB problems with 35 tasks (3 models, 4, 5, 7, and 12 stations), 70 tasks (5 models, 7, 10, 14, and 19 stations), 148 tasks (7 models, 10, 14, 21, and 29 stations). The results indicate that the proposed approach could generate a set of optimal nondominated solutions effectively satisfying the multiple objectives concurrently for mALB problems considering workers’ capabilities with the help of demand ratio-based cycle time reference.

Comparisons with existing moEA using same representation, crossover and mutation operator and different RWS operator, adaptive weight sum based fitness function, demonstrate that gp-siffGA could obtain both obviously better efficacy (about 66% improved in convergence performance) and efficiency (about 26% reduced in computation time) performances.

### 7.3 Multiobjective PPS Problem

The multiobjective PPS is to process a set of parts with operations on machines by selecting suitable manufacturing resources (machines, tools and tool access directions) and sequence the operations in manufacturing systems (MS) so as to rapidly and effectively determine a set of optimal solutions (schedules), which tells a production facility what to make, when, and on which equipment, to process the products effectively.

Therefore, issue 3 is efficient optimization of multiobjective PPS with minimization objectives for makespan, variation of workload of machine concurrently.

For improving both efficacy and efficiency performances, a fast and effective multiobjective evolutionary algorithm (FE-MOEA) is proposed.

The genetic representation consists of operation sequence, machine, tool and TAD vectors. The multiple vectors genetic representation is proposed to optimize the process planning and scheduling simultaneously.

As to fitness function, the gp-siff provides the obscure fitness value between nondominated and dominated individuals so that the nondominated ones might be lost in the evolving process. Therefore, a new Pareto dominating and dominated relationship-based (PDDR) fitness function is proposed. Moreover, PDDR has a clear time advan-
Conclusions
tage than most famous moEAs, NSGA-II and SPEA2, because it doesn’t consider the
dispersion performance. Additionally, individual locating around the central region
of Pareto font with bigger domination area will have better fitness value than the edge
points. Therefore, PDDR has also an advantage with the tendency converging toward
the central area of the Pareto front. Vector evaluated genetic algorithm (VEGA) has
an advantage with the edge region of the Pareto front duo to its selection only for one
objective. These above advantages are united tactfully in FE-MOEA. They preserve
both better convergence rate and dispersion performance.

The order-based crossover, resource-based mutation, problem-dependent local search
and BTS are adopted to improve the efficacy performance.

The proposed method is validated by 4 parts with 57 operations PPS problem. The
results indicate that the proposed approach could generate a set of optimal nondomi-
nated solutions effectively satisfying minimization of makespan and minimization of
variation of workload of machine concurrently for multiobjective PPS problem.

Complete comparisons indict that the efficacy performance of FE-MOEA is bet-
ter than the combination of gp-siff and VEGA (about 27% improved in convergence
performance, about 52% improved in dispersion performance) while the efficiency
performance is equivalent. Furthermore, the efficacy performance of FE-MOEA is
also better than NSGA-II and SPEA2 (about 87-94% improved in convergence perfor-
ance, 6-15% improved in dispersion performance), and the efficiency performance
is obviously better than theirs (about 23-49% reduced in computation time).

7.4 Multiobjective PPS Problem in DMS

Different from the multiobjectives PPS in single factory, the multiobjectives PPS in
DMS is to process a set of parts with operations on machines belonging to different
factories by assigning parts into factory, selecting suitable manufacturing resources
(machines, tools and TADs) and sequencing the operations so as to rapidly and ef-
effectively determine a set of optimal solutions (schedules), in which the precedence
constraints among operations can be satisfied and the corresponding objectives can be
achieved. This solutions tells a production facility what to make, when, and on which
equipment, to process the products effectively.

It leads issue 3 extending to issue 4, effective optimization of multiobjective PPS
in DMS with minimization objectives for maximum total processing time, maximum
variation of workload of machine at the same time.

For improving the efficacy performance and speeding up the efficiency perfor-
ance when solving the PPS problem in DMS with satisfying the two objectives con-
currently, fast multiobjective genetic algorithm with archive (fmoGA-A) approach was
proposed.

The multiple vector genetic representation, including part sequence, breakpoint,
operation priority, operation sequence, machine and tool as well as TAD vector, is
proposed. In the multiple vector, each PPS problem in each factory has been designed
as a whole. Crossover and mutation operators can be applied on these vectors to
optimize the PPS of multi-factory simultaneously.

A generalized Pareto-based scale-independent fitness function (gp-siff) was adopted
to evaluate the fitness function. It made the best use of Pareto dominance relationship
to evaluate individuals using a single measure, but it didn’t consider the rank and
other complex computation. The gp-siff could significantly speed up the computing times.

The elitism mechanism named archive for moGA was proposed to keep the “best” solutions from losing in evolving process. The individuals in archive with high probability could be selected into next generation. The archive mechanism could cover the selection bias to make the solutions converge the Pareto front. It could increase the quality of solutions. The replacement of archive according to gp-siff value also could enhance the quality of solutions. The gp-siff-based archive mechanism is used to improve the quality and speed.

The proposed method is validated by 2 factories, 4 parts PPS problem. The results indicate that the proposed approach could generate a set of optimal nondominated solutions effectively satisfying two minimization objectives for maximum total processing time, maximum variation of machine workload concurrently for multiobjective PPS problem in DMS.

These comparisons demonstrate the importance of PPS in DMS and indicate fmoGA-A was better than adaptive weight genetic algorithm (awGA) and gp-siffGA did on efficacy (59-82% improved). Comparing to NSGA-II and SPEA2, the efficacy (convergence) performance of fmoGA-A is not less than NSGA-II and SPEA2 did and the efficiency performance is obviously better than theirs (about 83-89% reduced in computation time).

7.5 Future Research

Generally, the proposed approaches would be particularly useful for solving multiobjective scheduling problems efficiently and effectively.

Although this study has provided a detailed study of the application of moEAs to solve multiobjective planning, scheduling and balancing problems, there is much improvement and expansion in future works.

According to smALB problem, if could find a heuristic method to determine the allocation of worker instead of assigning them randomly will improve the convergence speed. This problem-dependent heuristic method will provide the obviously help for solving large scale problems. For mALB problem, sequence problem need to be considered. If above heuristic method could deal with smALB, mALB also could improve by applying heuristic method. Heuristic algorithms are very important for good moEAs.

As PPS problem, construction of chromosome could be improved in the future. It is worthy of consideration for combining heuristic method to determine the selection of resources or combining learning approaches to select resources to improve the quality of solutions.

Some reproduction technologies with direction based methods are suggested to use in order to improve search ability and search speed.

Consideration of using local search method such as, changing orders of operations located in critical path, to search more solutions is planning to be considered.

Local search could use to improve the generate task/operation sequences by neighborhood search for ALB and PPS. However, it is necessary to mention that the balance between genetic search and local search is critical for effective search. In addition, the variation operators need to put more focus on the exploration because exploitation could be implemented by a local search method.
Moreover, more practical dynamic scheduling and rescheduling problems, just in time planning and scheduling problem as well as planning and scheduling for overall supply chain are worthy of being considered in the future.
Acknowledgments

During the past few years, spent in Kitakyusyu city for my Ph.D. is very important in my life. I learned patience, persistence, perspiration and positive, confident and optimistic attitude to life.

My deepest gratitude and recognition is to my advisor, Dr. Shigeru Fujimura, Professor at Waseda University, for unselfish giving of his time, invaluable guidance, and constructive comments toward the improvement of this study. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. He taught me how to express my ideas and showed me different ways to approach a research problem and the need to be persistent to accomplish any goal. His patience and support helped me overcome many crisis situations and finish this dissertation. I hope that one day I would become as good an advisor to my students as he has been to me.

Besides my advisor, I also would like to express my sincere thanks to Prof. Dr. Kotaro Hirasawa and Prof. Dr. Osamu Yoshie, the committee members, for their careful reviews, constructive advices and helpful suggestions in my work.

I would like to thank Prof. Dr. Mitsuo Gen, Prof. Dr. Keiichi Koyanagi whose meticulous suggestions and comments were of inestimable value to this study. Many thanks should be given to Prof. Dr. Xinjie Yu at Tsinghua University as a visiting scholar at IPS for one year. Studying and discussing on his new textbook, Introduction to Evolutionary Algorithms published at Springer make me learn more.

I would like to thank my colleagues and friends for the warm, supportive environment where they provided me.

My heartfelt gratitude especially to my families, my mother, Haoqing Zhen and my father, Shenxiu Zhang, and my sisters and brother. Their love and support provided me the energy to attain my study.

Lastly, I want to give my special gratitude to my girl friend, Zhan Qian with all my heart, who encouraged me to take up a Ph. D degree at Waseda University. Your love and help made this work possible.
List of Publications

Journals


International Conferences (with Review Process)


Others


Appendix A

Appendices to Chapters

Figure A.1: Gantt chart for the best compromised solution with \( w_1=1, w_2=0 \) and \( w_3=0 \) of smALB

Figure A.2: Gantt chart for the best compromised solution with \( w_1=0, w_2=1 \) and \( w_3=0 \) of smALB
Appendices to Chapters

Cycle Time: 101.00
Variation of Workload: 0.1626
Total Cost: 705.00

\[ w_1 = 0, w_2 = 0 \text{ and } w_3 = 1 \]

Figure A.3: Gantt chart for the best compromised solution with \( w_1 = 0, w_2 = 0 \) and, \( w_3 = 1 \) of smALB

Figure A.4: Precedence relationship graph of 35 tasks for smALB

Table A.1: Precedence relationship data of 35 tasks for smALB

<table>
<thead>
<tr>
<th>Task</th>
<th>Predecessor</th>
<th>Successor</th>
<th>Task</th>
<th>Predecessor</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
<td>Pre(j)</td>
<td>Suc(j)</td>
<td>j</td>
<td>Pre(j)</td>
<td>Suc(j)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2, 5, 7, 10, 12</td>
<td>19</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>20</td>
<td>17, 19</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>21</td>
<td>16, 20</td>
<td>22, 25, 30, 32</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>11</td>
<td>22</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>6</td>
<td>23</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>7, 8</td>
<td>24</td>
<td>23</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>14, 18</td>
<td>25</td>
<td>21</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>9</td>
<td>26</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>13</td>
<td>27</td>
<td>24, 26</td>
<td>28, 33, 34</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>14</td>
<td>28</td>
<td>15, 13, 27</td>
<td>29</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>28, 33</td>
<td>29</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>18</td>
<td>30</td>
<td>21</td>
<td>31</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>28, 33</td>
<td>31</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>7, 10</td>
<td>15</td>
<td>32</td>
<td>24, 31</td>
<td>33</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>16</td>
<td>33</td>
<td>14, 15, 25</td>
<td>32, 35</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>21</td>
<td>34</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>20</td>
<td>35</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>7, 12</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Task time / cost unit</td>
<td>Task</td>
<td>Task time / cost unit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------------</td>
<td>------</td>
<td>----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>40/60</td>
<td>35</td>
<td>5/95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>23/77</td>
<td></td>
<td>22/78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>19/81</td>
<td></td>
<td>19/81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2/98</td>
<td></td>
<td>2/98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>22/78</td>
<td></td>
<td>22/78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>19/81</td>
<td></td>
<td>19/81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5/95</td>
<td></td>
<td>5/95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5/95</td>
<td></td>
<td>5/95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>6/94</td>
<td></td>
<td>6/94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>22/78</td>
<td></td>
<td>22/78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>22/78</td>
<td></td>
<td>22/78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>23/77</td>
<td></td>
<td>23/77</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.2: Process time and worker cost data of 35 tasks for smALB
Figure A.5: Gantt chart for the best compromised solution with $w_1=1$, $w_2=0$ and $w_3=0$ of mALB

Figure A.6: Gantt chart for the best compromised solution with $w_1=0$, $w_2=1$ and $w_3=0$ of mALB
Figure A.7: Gantt chart for the best compromised solution with $w_1 = 0$, $w_2 = 0$ and $w_3 = 1$ of mALB
### Table A.4: Operation information for part 2 of PPS

<table>
<thead>
<tr>
<th>Feature-ID</th>
<th>Operation-ID</th>
<th>Operations</th>
<th>TAD candidates</th>
<th>Machine candidates</th>
<th>Tool candidates</th>
<th>Machining time (time unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>op1</td>
<td>Drilling +z, -z</td>
<td>m1, m2, m3, m4</td>
<td>t1</td>
<td></td>
<td>12, 10, 10, 7.5</td>
</tr>
<tr>
<td>f2</td>
<td>op2</td>
<td>Milling +x, +y, +z</td>
<td>m2, m3, m4</td>
<td>t2</td>
<td></td>
<td>20, 20, 15</td>
</tr>
<tr>
<td>f3</td>
<td>op3</td>
<td>Milling +x, +y, +z</td>
<td>m2, m3, m4</td>
<td>t3, t4</td>
<td></td>
<td>18, 18, 13.5</td>
</tr>
<tr>
<td>f4</td>
<td>op4</td>
<td>Milling +x, -x, +y, -y, +z</td>
<td>m2, m3, m4</td>
<td>t5 (t1, t2)</td>
<td></td>
<td>16, 16, 12</td>
</tr>
<tr>
<td>f5</td>
<td>op5</td>
<td>Milling +x, -x, -y, +z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f6</td>
<td>op6</td>
<td>Milling +x, -x, +y, -z</td>
<td>m2, m3, m4</td>
<td>t2</td>
<td></td>
<td>30, 25, 25, 18.75</td>
</tr>
<tr>
<td>f7</td>
<td>op7</td>
<td>Reraming +z, -z, +x, -y</td>
<td>m2, m3, m4</td>
<td>t9</td>
<td></td>
<td>25, 25, 18.75</td>
</tr>
<tr>
<td>f8</td>
<td>op8</td>
<td>Drilling +z, -z</td>
<td>m1, m2, m3, m4</td>
<td>t1</td>
<td></td>
<td>14, 14, 12, 9</td>
</tr>
<tr>
<td>f9</td>
<td>op9</td>
<td>Milling +x, -x, -y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f10</td>
<td>op10</td>
<td>Drilling +z, -z</td>
<td>m2, m3, m4, m5</td>
<td>t1</td>
<td></td>
<td>9, 6, 8, 6</td>
</tr>
<tr>
<td>f11</td>
<td>op11</td>
<td>Milling +y, +x, -x, -y</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>10, 10, 7.5</td>
</tr>
<tr>
<td>f12</td>
<td>op12</td>
<td>Milling +y, +x, -x, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>16, 16, 7.5</td>
</tr>
<tr>
<td>f13</td>
<td>op13</td>
<td>Milling +y, +x, -z, +z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>16, 16, 7.5</td>
</tr>
<tr>
<td>f14</td>
<td>op14</td>
<td>Milling +y, -y, +z, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>9, 6, 8, 6</td>
</tr>
<tr>
<td>f15</td>
<td>op15</td>
<td>Milling +y, -y, +z, -z</td>
<td>m2, m3, m4, m5</td>
<td>t6, t7, t8</td>
<td></td>
<td>36, 30, 30, 22.5</td>
</tr>
</tbody>
</table>

### Table A.5: Operation information for part 3 of PPS

<table>
<thead>
<tr>
<th>Feature-ID</th>
<th>Operation-ID</th>
<th>Operations</th>
<th>TAD candidates</th>
<th>Machine candidates</th>
<th>Tool candidates</th>
<th>Machining time (time unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>op1</td>
<td>Milling +x</td>
<td>m3, m5, m6</td>
<td>t6</td>
<td></td>
<td>20, 20, 15</td>
</tr>
<tr>
<td>f2</td>
<td>op2</td>
<td>Milling +y, +y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>20, 20, 15</td>
</tr>
<tr>
<td>f3</td>
<td>op3</td>
<td>Milling +y, +y, +z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f4</td>
<td>op4</td>
<td>Milling +y, +y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f5</td>
<td>op5</td>
<td>Milling +y, -y, +z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f6</td>
<td>op6</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f7</td>
<td>op7</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t7, t8, t9</td>
<td></td>
<td>15, 15, 11.25</td>
</tr>
<tr>
<td>f8</td>
<td>op8</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>25, 25, 25, 14.73</td>
</tr>
<tr>
<td>f9</td>
<td>op9</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>30, 25, 25, 18.75</td>
</tr>
<tr>
<td>f10</td>
<td>op10</td>
<td>Reraming +z, -z</td>
<td>m3, m5, m6</td>
<td>t9</td>
<td></td>
<td>20, 20, 15</td>
</tr>
<tr>
<td>f11</td>
<td>op11</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>20, 20, 15</td>
</tr>
<tr>
<td>f12</td>
<td>op12</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t6, t7, t8</td>
<td></td>
<td>9, 6, 8, 6</td>
</tr>
<tr>
<td>f13</td>
<td>op13</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t5</td>
<td></td>
<td>8, 8, 6</td>
</tr>
<tr>
<td>f14</td>
<td>op14</td>
<td>Milling +y, -y, -z</td>
<td>m2, m3, m4</td>
<td>t5</td>
<td></td>
<td>6, 6, 5, 5, 1.25</td>
</tr>
</tbody>
</table>
### Table A.6: Operation information for part 4 of PPS

<table>
<thead>
<tr>
<th>Feature-ID</th>
<th>Operation-ID</th>
<th>Operations</th>
<th>TAD candidates</th>
<th>Machine candidates</th>
<th>Tool candidates</th>
<th>Machining time (time unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>op1</td>
<td>Milling</td>
<td>-y</td>
<td>m2</td>
<td>t6, t9</td>
<td>12,15</td>
</tr>
<tr>
<td>f2</td>
<td>op2</td>
<td>Milling</td>
<td>-y</td>
<td>m2</td>
<td>t9, t10</td>
<td>21,18</td>
</tr>
<tr>
<td>f3</td>
<td>op3</td>
<td>Milling</td>
<td>-x</td>
<td>m2</td>
<td>t3</td>
<td>18,25</td>
</tr>
<tr>
<td>f4</td>
<td>op4</td>
<td>Milling</td>
<td>-z, -x</td>
<td>m3</td>
<td>T4</td>
<td>27,25</td>
</tr>
<tr>
<td>f5</td>
<td>op5</td>
<td>Drilling</td>
<td>-x</td>
<td>m2, m3</td>
<td>c2</td>
<td>20,25,18</td>
</tr>
<tr>
<td>f6</td>
<td>op6</td>
<td>Milling</td>
<td>-x</td>
<td>m2</td>
<td>t3</td>
<td>18,15</td>
</tr>
<tr>
<td>f7</td>
<td>op7</td>
<td>Drilling</td>
<td>+z, +x</td>
<td>m4, m2</td>
<td>t2</td>
<td>20,15,18</td>
</tr>
</tbody>
</table>

### Table A.7: Precedence relationship for part 1 of PPS

<table>
<thead>
<tr>
<th>Operation</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>op1</td>
<td>op2, op3, op5, op6, op7, op8, op9, op10, op11, op12, op13, op14, op15, op16, op17, op18, op19, op20</td>
</tr>
<tr>
<td>op2</td>
<td>op12, op13, op14, op15, op16</td>
</tr>
<tr>
<td>op3</td>
<td>--</td>
</tr>
<tr>
<td>op4</td>
<td>--</td>
</tr>
<tr>
<td>op5</td>
<td>op4, op7</td>
</tr>
<tr>
<td>op6</td>
<td>op2, op11, op14</td>
</tr>
<tr>
<td>op7</td>
<td>op6, op8, op10</td>
</tr>
<tr>
<td>op8</td>
<td>op10, op11</td>
</tr>
<tr>
<td>op9</td>
<td>op10</td>
</tr>
<tr>
<td>op10</td>
<td>op12, op13, op14</td>
</tr>
<tr>
<td>op11</td>
<td>op12, op13, op14</td>
</tr>
<tr>
<td>op12</td>
<td>op13, op14, op15, op16, op18, op21</td>
</tr>
<tr>
<td>op13</td>
<td>op4, op11, op14, op15, op16, op19, op20</td>
</tr>
<tr>
<td>op14</td>
<td>op4, op11, op14, op15, op16, op19, op20</td>
</tr>
<tr>
<td>op15</td>
<td>op14, op16, op17, op20</td>
</tr>
<tr>
<td>op16</td>
<td>op17</td>
</tr>
<tr>
<td>op17</td>
<td>--</td>
</tr>
<tr>
<td>op18</td>
<td>op14, op17</td>
</tr>
<tr>
<td>op19</td>
<td>op20</td>
</tr>
<tr>
<td>op20</td>
<td>--</td>
</tr>
</tbody>
</table>
Table A.8: Precedence relationship for part 2 of PPS

<table>
<thead>
<tr>
<th>Operation</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>op1</td>
<td>op2</td>
</tr>
<tr>
<td>op2</td>
<td></td>
</tr>
<tr>
<td>op3</td>
<td>op4</td>
</tr>
<tr>
<td>op4</td>
<td></td>
</tr>
<tr>
<td>op5</td>
<td></td>
</tr>
<tr>
<td>op6</td>
<td>op7, op8</td>
</tr>
<tr>
<td>op7</td>
<td>op8</td>
</tr>
<tr>
<td>op8</td>
<td></td>
</tr>
<tr>
<td>op9</td>
<td>op10</td>
</tr>
<tr>
<td>op10</td>
<td></td>
</tr>
<tr>
<td>op11</td>
<td></td>
</tr>
<tr>
<td>op12</td>
<td>op13, op14</td>
</tr>
<tr>
<td>op13</td>
<td></td>
</tr>
<tr>
<td>op14</td>
<td></td>
</tr>
<tr>
<td>op15</td>
<td></td>
</tr>
<tr>
<td>op16</td>
<td>op1, op2, op3, op4, op5, op6, op7, op8, op9, op10, op11, op12, op13, op14, op15, op16</td>
</tr>
</tbody>
</table>

Table A.9: Precedence relationship for part 3 of PPS

<table>
<thead>
<tr>
<th>Operation</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>op1</td>
<td>op2, op3, op4, op5, op6, op7, op8, op9, op10, op11, op12, op13, op14</td>
</tr>
<tr>
<td>op2</td>
<td>op3, op4, op5, op6, op7, op8, op9, op10, op11, op12, op13, op14</td>
</tr>
<tr>
<td>op3</td>
<td>op12, op13, op14</td>
</tr>
<tr>
<td>op4</td>
<td>op12, op13, op14</td>
</tr>
<tr>
<td>op5</td>
<td>op12, op13, op14</td>
</tr>
<tr>
<td>op6</td>
<td>op12, op13, op14</td>
</tr>
<tr>
<td>op7</td>
<td></td>
</tr>
<tr>
<td>op8</td>
<td></td>
</tr>
<tr>
<td>op9</td>
<td>op5, op6, op13</td>
</tr>
<tr>
<td>op10</td>
<td>op5, op6, op13</td>
</tr>
<tr>
<td>op11</td>
<td>op5, op6, op13</td>
</tr>
<tr>
<td>op12</td>
<td>op11</td>
</tr>
<tr>
<td>op13</td>
<td></td>
</tr>
<tr>
<td>op14</td>
<td></td>
</tr>
</tbody>
</table>

Table A.10: Precedence relationship for part 4 of PPS

<table>
<thead>
<tr>
<th>Operation</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>op1</td>
<td>op5, op4</td>
</tr>
<tr>
<td>op2</td>
<td>op4, op6, op7</td>
</tr>
<tr>
<td>op3</td>
<td>op5, op4</td>
</tr>
<tr>
<td>op4</td>
<td></td>
</tr>
<tr>
<td>op5</td>
<td></td>
</tr>
<tr>
<td>op6</td>
<td>op4, op7</td>
</tr>
<tr>
<td>op7</td>
<td></td>
</tr>
</tbody>
</table>
Figure A.8: Precedence relationship graph for 4 parts of PPS
# List of Figures

1.1 Precedence relationship graph of an example for ALB ........................................... 4
1.2 Single-model assembly line ..................................................................................... 5
1.3 Multi-model assembly line ...................................................................................... 6
1.4 Mixed-model assembly line ..................................................................................... 6
1.5 Serial assembly line ............................................................................................... 7
1.6 U-shaped assembly line .......................................................................................... 7
1.7 Parallelism assembly line ....................................................................................... 7
1.8 A raw prismatic part for PP .................................................................................... 9
1.9 Precedence relationship of PP problem with 7 operations ........................................ 10
2.1 Evolving process in one generation of VEGA ......................................................... 27
2.2 Evolving process in one generation of NSGA-II ..................................................... 29
2.3 Evolving process in one generation of SPEA2 ....................................................... 29
2.4 Number of obtained solutions ................................................................................ 32
2.5 Error ratio ............................................................................................................... 32
2.6 Ratio of nondominated solutions ........................................................................... 33
2.7 Coverage $C(S_1, S_2)$ and $C(S_2, S_1)$ ................................................................. 33
2.8 Hypervolume $HV(S_1)$ and $HV(S_2)$ .................................................................. 34
2.9 Generational distance ............................................................................................. 35
2.10 Spacing .................................................................................................................. 36
3.1 Precedence relationship graph of the smALB model .............................................. 39
3.2 Description of the smALB model ............................................................................ 40
3.3 Task priority and worker allocation vector of the example for smALB ................. 44
3.4 Procedure of encoding for task priority vector of smALB .................................... 45
3.5 Procedure of encoding for worker allocation vector of smALB .......................... 46
3.6 Procedure of decoding of task sequence for smALB ............................................ 46
3.7 Task sequence vector of the example for smALB .................................................. 46
3.8 Procedure of decoding of breakpoint for smALB ................................................ 48
3.9 Breakpoint vector of the example for smALB ....................................................... 48
3.10 Phenotype chromosome of the example for smALB .......................................... 49
3.11 Illustration of arithmetic crossover for task priority vector for smALB .............. 50
3.12 Procedure of weight mapped crossover for worker allocation vector for smALB ........................................................................................................................................ 51
3.13 Illustration of weight mapped crossover for worker allocation vector for smALB ........................................................................................................................................ 51
3.14 Illustration of swap mutation for smALB ............................................................. 52
3.15 Procedure of roulette wheel selection for smALB .............................................. 53
3.16 Gantt chart for the best compromised solution with $w_1=0.5$, $w_2=0.2$ and, $w_3=0.3$ of smALB .... 54
3.17 Gantt chart of the best compromise solution (3 products) with $w_1=0.5$, $w_2=0.2$ and, $w_3=0.3$ of smALB .... 55
3.18 Comparison of the generational distance about priGA and rkGA for smALB .... 56
3.19 Comparison of means of CPU times by rkGA and priGA of smALB .... 57

4.1 Description of the mALB model .... 61
4.2 Precedence relationship graph of model 1 for the mALB .... 61
4.3 Precedence relationship graph of model 2 for the mALB .... 62
4.4 Comparison of demand ratio-based cycle time and average cycle time for the mALB .... 66
4.5 Precedence relationship graph of combined model for the mALB .... 69
4.6 Task priority and worker allocation vector of the example for mALB .... 70
4.7 Task sequence vector of the example for mALB .... 70
4.8 Breakpoint vector of the example for mALB .... 71
4.9 Phenotype chromosome of the example for mALB .... 72
4.10 An example of gp-siff .... 73
4.11 Gantt chart for the best compromised solution with $w_1=0.5$, $w_2=0.2$ and, $w_3=0.3$ of mALB .... 74
4.12 Gantt chart of the best compromise solution (5 products) with $w_1=0.5$, $w_2=0.2$ and, $w_3=0.3$ of mALB .... 75
4.13 Box plots based on the coverage for mALB .... 77
4.14 Comparison of means of CPU times measures by gp-siffGA and rkGA of mALB .... 79

5.1 Illustration of PPS problem .... 83
5.2 The comparisons of GP-SIFF and PDDR .... 87
5.3 The comparison process of PDDR .... 88
5.4 The description of FE-MOEA .... 89
5.5 The evolving process in one generation of FE-MOEA .... 89
5.6 A chromosome of the example .... 90
5.7 The description of 4 parts of PPS .... 93
5.8 Pareto set by FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2 .... 94
5.9 50% attainment surface by FE-MOEA, GP-SIFFGA, NSGA-II and SPEA2 .... 95
5.10 C, GD, HV, SP and CPU times by FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2 .... 97

6.1 Example of PPS problem in DMS .... 104
6.2 Evolving process in one generation of fmoGA-A for PPS in DMS .... 109
6.3 Description of 4 parts of PPS in DMS .... 110
6.4 Information for 2 factories of PPS in DMS .... 110
6.5 Part sequence vector ($v_1$) of the example of PPS in DMS .... 111
6.6 Breakpoint vector ($v_2$) of the example of PPS in DMS .... 112
6.7 Operation priority vector ($v_3$) of the example of PPS in DMS .... 112
6.8 Priority vector for part 1 of the example of PPS in DMS .... 112
6.9 Priority vector for part 2, 3, and 4 of the example of PPS in DMS .... 113
6.10 Operation sequence vector ($v_4$) for 4 parts of the example of PPS in DMS .... 113
6.11 Machine and tool vector \((v_5)\) for 4 parts of the example of PPS in DMS . 113
6.12 Tool access direction vector \((v_6)\) for 4 parts of the example of PPS in DMS . 114
6.13 Comparison of obtained Pareto set by fmoGA-A, awGA and fmoGA of PPS in DMS . 116
6.14 Comparison of generational distance, hypervolume and CPU times measures by fmoGA-A, awGA and fmoGA of PPS in DMS . 116
6.15 Comparison of obtained Pareto set by fmoGA-A, NSGA-II and SPEA2 of PPS in DMS . 117
6.16 Comparison of generational distance, hypervolume and CPU times measures by fmoGA-A, NSGA-II, and SPEA2 of PPS in DMS . 119
6.17 Comparison of obtained Pareto set by fmoGA-A, NSGA-II, and SPEA2 of PPS in DMS . 117
6.18 Comparison of mean of generational distance, hypervolume and CPU times measures by fmoGA-A, NSGA-II, and SPEA2 of PPS in DMS . 119
A.1 Gantt chart for the best compromised solution with \(w_1=1, w_2=0\) and, \(w_3=0\) of smALB . 131
A.2 Gantt chart for the best compromised solution with \(w_1=0, w_2=1\) and, \(w_3=0\) of smALB . 131
A.3 Gantt chart for the best compromised solution with \(w_1=0, w_2=0\) and, \(w_3=1\) of smALB . 132
A.4 Precedence relationship graph of 35 tasks for smALB . 132
A.5 Gantt chart for the best compromised solution with \(w_1=1, w_2=0\) and, \(w_3=0\) of mALB . 134
A.6 Gantt chart for the best compromised solution with \(w_1=0, w_2=1\) and, \(w_3=0\) of mALB . 134
A.7 Gantt chart for the best compromised solution with \(w_1=0, w_2=0\) and, \(w_3=1\) of mALB . 135
A.8 Precedence relationship graph for 4 parts of PPS . 139
List of Tables

1.1 Data set of an example for ALB ........................................ 3
1.2 Data set of PP problem with 7 operations .......................... 10

3.1 Data set of the smALB model ........................................... 39
3.2 Trace table of decoding of task sequence for smALB .......... 47
3.3 Trace table of decoding of breakpoint for smALB ............... 48
3.4 Comparison of the generational distance (mean and standard deviation) about priGA and rkGA for smALB .......................... 56
3.5 Comparison of CPU time (mean and standard deviation) about priGA and rkGA for smALB .............................. 57

4.1 Data set of model 1 for mALB ......................................... 62
4.2 Data set of model 2 for mALB ......................................... 62
4.3 Data set of cost for mALB .............................................. 63
4.4 Data set of combined for mALB ........................................ 69
4.5 Information for three benchmark problems for mALB ....... 75
4.6 Comparison of the Coverage (mean and standard deviation) about gp-siffGA and rkGA for smALB .......................... 78
4.7 Comparison of the CPU time (mean and standard deviation) about gp-siffGA and rkGA for smALB .............................. 79

5.1 Operation information of part 1 to part 4 ............................ 84
5.2 Comparison of the coverage (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2 .......... 95
5.3 Comparison of the generational distance (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2 ........... 96
5.4 Comparison of the hypervolume (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2 .......... 96
5.5 Comparison of the spacing (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2 .......... 96
5.6 Comparison of the CPU time (mean and standard deviation) about FE-MOEA, GP-SIFFGA, NSGA-II, and SPEA2 .......... 99
5.7 Comparison of time complexity ....................................... 100

6.1 Comparison of the generational distance (mean and standard deviation) about fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2 .......... 117
6.2 Comparison of the hypervolume (mean and standard deviation) about fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2 .......... 118
<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Comparison of the CPU time (mean and standard deviation) about fmoEA-A, awGA, fmoGA, NSGA-II and SPEA2</td>
<td>118</td>
</tr>
<tr>
<td>6.4</td>
<td>$t$ test result of fmoGA-A, NSGA-II and SPEA2 of PPS in DMS</td>
<td>120</td>
</tr>
<tr>
<td>A.1</td>
<td>Precedence relationship data of 35 tasks for smALB</td>
<td>132</td>
</tr>
<tr>
<td>A.2</td>
<td>Process time and worker cost data of 35 tasks for smALB</td>
<td>133</td>
</tr>
<tr>
<td>A.3</td>
<td>Operation information for part 1 of PPS</td>
<td>135</td>
</tr>
<tr>
<td>A.4</td>
<td>Operation information for part 2 of PPS</td>
<td>136</td>
</tr>
<tr>
<td>A.5</td>
<td>Operation information for part 3 of PPS</td>
<td>136</td>
</tr>
<tr>
<td>A.6</td>
<td>Operation information for part 4 of PPS</td>
<td>137</td>
</tr>
<tr>
<td>A.7</td>
<td>Precedence relationship for part 1 of PPS</td>
<td>137</td>
</tr>
<tr>
<td>A.8</td>
<td>Precedence relationship for part 2 of PPS</td>
<td>138</td>
</tr>
<tr>
<td>A.9</td>
<td>Precedence relationship for part 3 of PPS</td>
<td>138</td>
</tr>
<tr>
<td>A.10</td>
<td>Precedence relationship for part 4 of PPS</td>
<td>138</td>
</tr>
</tbody>
</table>


planning for job shop machining. IEEE Transactions on Evolutionary Computation, 
1, 278–289.


Part I. Feature creation and selection of operations and tools. Journal of Materials 

swarm optimisation in integrated process planning and scheduling. Robotics and 

ASTME paper, 737.

Planning and Scheduling: A State-of-the-Art Survey. IEEE Transactions on Sys-
tems, Man, and Cybernetics, Part C: Applications and Reviews, 36, 563–577.


42, 739–751.

rithm approach. Assembly and Task Planning, 1999.(ISATP’99) Proceedings of the 

[40] Usher, J. and Bowden, R. (1996) The application of genetic algorithms to op-
eration sequencing for use in computer-aided process planning. Computers & 
Industrial Engineering, 30, 999–1013.


ments in different levels and stages of machining using one general ANN-based 


planning in feature based, open-architecture manufacturing. Proceedings of 
ASME DETC, vol. 96.


Index

ACO, 8, 20
adaptive weight genetic algorithm, 26
AI, 19
AL, 1
ALB, 1, 3
ant colony optimization, 8, 20
artificial intelligence, 19
assembly line, 1
assembly line balancing, 1, 8
Type-1, 7
Type-2, 7
Type-3, 7
Type-4, 7
Type-5, 7
Type-E, 7
Type-F, 7
average distance, 34
awGA, 26

binary index, 31

CAD, 14
CAM, 14
CAPP, 14
code, 25
combinatorial optimization problem, 1, 3
computation time, 2
computer-aided design, 14
computer-aided manufacturing, 14
computer-aided process planning, 14
convergence, 2
cover, 24
coverage, 32, 33
coverage difference, 34
criterion space, 24
crowding distance, 28
cycle time, 2, 3
DE, 20
decision space, 24
decision vector, 24
density, 30
differential evolution, 20
dispersion, 2
distributed manufacturing system, 2, 17
DMS, 2, 17
dominance, 24
dynamic programming, 8

EA, 19
EAs, 25
EC, 19, 25
efficacy, 17
efficiency, 17
efficient solution, 25
EI, 25
EMO, 25
EP, 20
equivalent, 24
error ratio, 31
ES, 20
evolution strategy, 20
evolutionary algorithms, 19, 25
evolutionary computation, 19, 25
evolutionary intelligence, 25
evolutionary multiobjective optimization, 25
evolutionary programming, 20

fitness space, 24
fitness value, 25
flexible manufacturing system, 16
FMS, 16

GA, 8, 20
gALB, 6
Gaussian adaption, 20
general assembly line balancing, 6
generating approach, 26
generational distance, 34
generative, 14
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>genetic algorithm</td>
<td>20</td>
</tr>
<tr>
<td>genetic algorithms</td>
<td>8</td>
</tr>
<tr>
<td>genetic programming</td>
<td>20</td>
</tr>
<tr>
<td>GP</td>
<td>20</td>
</tr>
<tr>
<td>harmony search</td>
<td>20</td>
</tr>
<tr>
<td>hypervolume</td>
<td>33</td>
</tr>
<tr>
<td>hypervolume ratio</td>
<td>34</td>
</tr>
<tr>
<td>idle time</td>
<td>3</td>
</tr>
<tr>
<td>individual</td>
<td>8</td>
</tr>
<tr>
<td>line efficiency</td>
<td>2</td>
</tr>
<tr>
<td>linear programming</td>
<td>8</td>
</tr>
<tr>
<td>mAL</td>
<td>2</td>
</tr>
<tr>
<td>mALB</td>
<td>2, 5</td>
</tr>
<tr>
<td>manufacturing system</td>
<td>11</td>
</tr>
<tr>
<td>maximum extreme point</td>
<td>26</td>
</tr>
<tr>
<td>meta-heuristic</td>
<td>19</td>
</tr>
<tr>
<td>minimum extreme point</td>
<td>26</td>
</tr>
<tr>
<td>mixed-model assembly line balancing</td>
<td>2, 5</td>
</tr>
<tr>
<td>mixed-model assembly lines</td>
<td>2</td>
</tr>
<tr>
<td>model</td>
<td>2</td>
</tr>
<tr>
<td>moEAs</td>
<td>1, 25</td>
</tr>
<tr>
<td>moOP</td>
<td>2, 23</td>
</tr>
<tr>
<td>MS</td>
<td>1</td>
</tr>
<tr>
<td>muALB</td>
<td>5</td>
</tr>
<tr>
<td>multi-model assembly line balancing</td>
<td>5</td>
</tr>
<tr>
<td>multiobjective evolutionary algorithms</td>
<td>1, 25</td>
</tr>
<tr>
<td>multiobjective optimization problem</td>
<td>2</td>
</tr>
<tr>
<td>mutation</td>
<td>19</td>
</tr>
<tr>
<td>nondominated solution</td>
<td>24</td>
</tr>
<tr>
<td>nondominated solutions</td>
<td>2</td>
</tr>
<tr>
<td>nondominated sorting genetic algorithm</td>
<td>28</td>
</tr>
<tr>
<td>Nondominated Sorting Genetic Algorithm II</td>
<td>26</td>
</tr>
<tr>
<td>noninferior solution</td>
<td>25</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>26, 28</td>
</tr>
<tr>
<td>number of obtained solutions</td>
<td>31</td>
</tr>
<tr>
<td>objective space</td>
<td>24</td>
</tr>
<tr>
<td>objective vector</td>
<td>24</td>
</tr>
<tr>
<td>operation</td>
<td>2</td>
</tr>
<tr>
<td>operation sequence</td>
<td>10</td>
</tr>
<tr>
<td>Parallelism assembly line</td>
<td>6</td>
</tr>
<tr>
<td>parameter space</td>
<td>24</td>
</tr>
<tr>
<td>parameter vector</td>
<td>24</td>
</tr>
<tr>
<td>Pareto front</td>
<td>25</td>
</tr>
<tr>
<td>Pareto frontier</td>
<td>25</td>
</tr>
<tr>
<td>Pareto optimal front</td>
<td>25</td>
</tr>
<tr>
<td>Pareto optimal set</td>
<td>25</td>
</tr>
<tr>
<td>Pareto optimal solution</td>
<td>24</td>
</tr>
<tr>
<td>Pareto optimal solutions</td>
<td>2</td>
</tr>
<tr>
<td>Pareto set</td>
<td>25</td>
</tr>
<tr>
<td>Pareto solution set</td>
<td>25</td>
</tr>
<tr>
<td>part</td>
<td>2</td>
</tr>
<tr>
<td>particle swarm algorithm</td>
<td>8</td>
</tr>
<tr>
<td>particle swarm optimization</td>
<td>20</td>
</tr>
<tr>
<td>PDDR</td>
<td>82</td>
</tr>
<tr>
<td>performance indices</td>
<td>31</td>
</tr>
<tr>
<td>PI</td>
<td>31</td>
</tr>
<tr>
<td>population</td>
<td>25</td>
</tr>
<tr>
<td>positive ideal point</td>
<td>26</td>
</tr>
<tr>
<td>PP</td>
<td>2</td>
</tr>
<tr>
<td>PPS</td>
<td>1, 14</td>
</tr>
<tr>
<td>precedence relationship</td>
<td>2, 3</td>
</tr>
<tr>
<td>preference-based approach</td>
<td>26</td>
</tr>
<tr>
<td>process plan</td>
<td>9</td>
</tr>
<tr>
<td>process planning</td>
<td>9</td>
</tr>
<tr>
<td>process planning and scheduling</td>
<td>1</td>
</tr>
<tr>
<td>production cost</td>
<td>2</td>
</tr>
<tr>
<td>PSO</td>
<td>8, 20</td>
</tr>
<tr>
<td>ratio of nondominated solutions</td>
<td>82</td>
</tr>
<tr>
<td>raw fitness</td>
<td>30</td>
</tr>
<tr>
<td>recombination</td>
<td>19</td>
</tr>
<tr>
<td>reference set</td>
<td>31</td>
</tr>
<tr>
<td>reproduction</td>
<td>19</td>
</tr>
<tr>
<td>roulette wheel selection</td>
<td>52</td>
</tr>
<tr>
<td>RWS</td>
<td>52</td>
</tr>
<tr>
<td>SA</td>
<td>8</td>
</tr>
<tr>
<td>sALB</td>
<td>6</td>
</tr>
<tr>
<td>selection</td>
<td>19</td>
</tr>
<tr>
<td>Serial assembly line</td>
<td>6</td>
</tr>
<tr>
<td>set-up</td>
<td>10</td>
</tr>
<tr>
<td>simple assembly line balancing</td>
<td>6</td>
</tr>
<tr>
<td>simulated annealing</td>
<td>8</td>
</tr>
<tr>
<td>single-model assembly line balancing</td>
<td>5</td>
</tr>
<tr>
<td>smALB</td>
<td>5</td>
</tr>
</tbody>
</table>
Index

spacing, 35
SPEA2, 26, 29
station, 2
strength, 29
strength Pareto evolution algorithm, 29
Strength Pareto Evolutionary Algorithm
2, 26
strongly dominate, 24

tabu search, 8
TAD, 10
task, 2
task time, 2
tool access direction, 10
total processing time, 13
TS, 8

U-shaped assembly line, 6
unary index, 31

variant, 14
Vector Evaluated Genetic Algorithm, 26
vector evaluated genetic algorithm, 27
VEGA, 26, 27

weakly dominate, 24
workload, 2
workstations, 3