Models and Solution Procedures in the Design and Scheduling of Manufacturing Systems with Distributed Layouts

by

Seyedfarhad Shafigh

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements

for the degree of

Doctor of Philosophy

in

Engineering

Guelph, Ontario, Canada

ABSTRACT

MODELS AND SOLUTION PROCEDURES IN THE DESIGN AND SCHEDULING OF MANUFACTURING SYSTEMS WITH DISTRIBUTED LAYOUTS

Seyedfarhad Shafigh **Advisor: Professor F.M. Defersha**

University of Guelph, 2015 **Co - advisor: Professor S.E. Moussa**

Numerous studies have been conducted to design facility layouts since the early 1950s. The majority of these studies have primarily focused on product layout, functional layout, cellular layout or their variants. Recent trend in manufacturing systems literature establishes the consensus that these conventional configurations do not meet the needs of today's multi-product enterprises working in dynamic environment. A promising approach to address changes in the production environment is to build facility layouts that can easily adapt to volatilities. Distributed layouts are among such facilities enabling industries to address volatilities and uncertainties.

This thesis addresses two distinct problems in facility design and scheduling for manufacturing firms operating in volatile environment and producing the multiple batches of products. In regards to the facility layout problem, a new comprehensive mathematical model that integrates layout configuration and production planning in the design of dynamic distributed layouts is formulated. The model incorporates a number of important manufacturing attributes such as demand fluctuation, system reconfiguration, lot splitting, work load balancing, alternative routings, machine capability and tooling requirements. In addition, the

model allows the optimization of several cost elements in an integrated manner. These include material handling, machine relocation, setup, inventory carrying, in-house production and subcontracting costs. With respect to the scheduling problem, a mathematical formulation for scheduling of manufacturing systems with distributed layouts is developed. The objective of scheduling model is the minimization of the weighted sum of makespan and total traveling distance by the products. Thus on one hand, the problem is to find a schedule of operations on machines (the sequence and starting times of the various operations) which minimizes the overall finishing time or makespan. On the other hand, the problem is to find assignment of jobs to the machines such that total distance traveled by parts is minimized.

Optimal solutions for the proposed mathematical models can only be found for small size problems due to NP-complexity. To solve both models for larger-size problems, two hybrids metaheuristics (linear programming embedded a metaheuristic) for solving the facility design model and a genetic algorithm for the scheduling model have been developed. All proposed algorithms are thoroughly examined with an emphasis on solution convergence, solution quality and algorithm robustness. For both cases, we provide numerical results to support various managerial insights. In particular in facility design problem, we draw a managerial insight as to how high product variety and high volatility in the production environment can be accommodated without harm to operational efficiency or cost. Similarly in the scheduling study, we show that linking scheduling and material handling performance can contribute to the development of accurate models to obtain a schedule that can also greatly enhance system performance.

Keywords: *Distributed Layout; Scheduling; Dynamic Reconfiguration; Hybrid metaheuristic Algorithm; Linear Programming; Multi-objective Optimization.*

Dedicated to my wife, Sameneh

ACKNOWLEDGEMENTS

I wish to express my gratitude to Professor Fantahun Defersha who guided me with constant encouragement, intellectual insights and critical advices which made this thesis possible. I would not have been able to join the PhD program at the University of Guelph without the financial support from the Growth Fund of the College of Physical and Engineering Science of this university and NSERC-Discovery funding thought Professor Defersha.

I am grateful to Professor Soha Eid Moussa for her patience in correcting my thesis and her useful comments on my research. I would like to thank the School of Engineering for providing comfortable learning and wide research environment.

I also wish to thank my Ph.D. advisory committee members, Professor Marwan Hassan and Professor Loong-Tak Lim, for their valuable comments and suggestion during my research.

I also wish to thank my wife, Samaneh Aminian for her encouragement, enthusiasm and emotional support, without which I would not have achieved this degree.

I would like to sincerely acknowledge my parents for their consistent love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

T	Number of equal planning periods where planning periods are indexed by $t = 1, 2, ..., T$.
P	Number of products where products are indexed by $p = 1, 2, ..., P$.
O_p	Number of operation required by product p where operation are indexed by $o = 1, 2, ..., O_p$.
N_p	Maximum number of sublots of product p in a given time period where production sublots are indexed by $n = 1, 2, ..., N_p$.
M	Number of machine in the manufacturing facility where machine are indexed by $m = 1, 2, ..., M$.
R	Number of resource elements in the manufacturing facility where resource elements are indexed by $r = 1, 2, ..., R$.
L	Number of location where machines are installed. Locations are indexed by $l = 1, 2, ..., L$.
J	Number of groups of machines with similar functionality where groups are indexed by $j = 1, 2, ..., J$.
C	Length of a planning period in terms of available working hours.
$D_{p,t}$	Demand quantity for product p in time period t.
Θ_p	Unit cost of producing product p in-house (not including setup).
$\hat{\Theta}_p$	Unit cost of outsourcing product p.
H_p	Unit inventory holding cost per period for product p.

F_p Material handling cost per unit distance for one unit of product p.

$U_{o,p}$ Unit processing time for operation o of product p.

$A_{r,m}$ A binary data which equal to 1 if resource element r is available on machine m; 0 otherwise.

$B_{r,o,p}$ A binary data which equal to 1 if resource element r is required by operation o of product p; 0 otherwise. An operations requires only a single resource element and machines having this resource element are considered as alternative routing for this operation.

$K_{o,p,m}$ A binary data which equal to 1 if operation o of product p can be processed on machine m; 0 otherwise. $K_{o,p,m} = \sum_{r=1}^{R}(A_{r,m} \times B_{r,o,p})$.

$E_{l,l'}$ Distance between location l and l'.

G_m Relocation cost per unit distance for machine m.

S_p Setup cost for processing a sublot of product p.

Ω Large positive number.

$v_{p,t}$ Production lot size of product p in time period t.

$b_{n,p,t}$ The size of the n^{th} sublot of product p in time period t.

$\hat{v}_{p,t}$ Volume of product p outsourced in time period t.

$\delta_{o,n,p,m,t}$ The time elapsed in processing operation o of the n^{th} sublot of product p on machine m in time period t.

$h_{p,t}$ Inventory level of product p at the beginning of period t.

$d_{o,n,p,t}$	Distance between the locations where operation o and $o+1$ of n^{th} sublot of product p are processed multiplied by the sublot size $b_{n,p,t}$ in time period t.
$e_{m,t}$	Distance between the location of machine m in period $t-1$ and its location in period t.
$\alpha_{m,l,t}$	A binary variable equal to 1 if machine m is located at location l in time period t; 0 otherwise.
$\gamma_{o,n,p,m,t}$	A binary variable equal to 1 if operation o of the n^{th} sublot of product p is processed by machine m in time period t.
$y_{n,p,t}$	A binary variable equal to 1 if n^{th} sublot of product p is created and processed in time period t.
S	Index of search path, $s = 1, 2, ..., S$ where S is the number of paths followed.
n	Iteration counter, $n = 1, 2, ..., N$ where N is the maximum number of iterations in each search path.
$X_{n,s}$	The solution at the n^{th} iteration along the s^{th} search path.
α	Cooling schedule exponent
r	Index for the temperature levels in the cooling schedule.
T_r	Temperature at the r^{th} level, $T_r = \alpha \times T_{r-1} = \alpha^r \times T_0$
Q	Number of iterations to be performed in each search path at each temperature level.
BI	Best feasible individual so far found.

$Phase$	The current phase of the search which equals to 1 for the static phase or 2 for the dynamic phase.
C_{phase}	The number of iterations to be performed by each search path before the search phase is changed from $Phase = 1$ to $Phase = 2$.
N	Population size
c	Index for a chromosome in a given population
k	Generation counter
$MaxGen$	Maximum generations
$Phase$	an indicator number which equals to 1 for the static phase or 2 for the dynamic phase
g_{phase}	Generation at which the value of $Phase$ should be set equal to 2 if it were not previously set to this value by other conditions
WoI	Number of successive population rejuvenations counted without any improvement of the best individual so far found
WoI_{max1}	Maximum value of WoI at which point the second phase is to be entered if $Phase$ was equal to 1
WoI_{max2}	Maximum value of WoI in the second phase at which point the search will be terminated.
R_m	Maximum number of production runs of machine m where production runs are indexed by r or $u = 1, 2, ..., R_m$; The assignment of operations to production runs determines their sequence;
$P_{o,j,m}$	A binary data equal to 1 if operation o of job j can be processed on machine m, 0 otherwise;

$A_{o,j}$	A binary data equal to 1 if the setup of operation o of job j is attached (non-anticipatory), or 0 if this setup is detached (anticipatory);
Ω	Large positive number.
$c_{o,j,m}$	Completion time of operation o of job j on machine m;
$\widehat{c}_{r,m}$	Completion time of the r^{th} run of machine m;
c_{max}	Makespan of the schedule
$x_{r,m,o,j}$	Binary variable which takes the value 1 if the r^{th} run on machine m is for operation o of job j, 0 otherwise;
$z_{r,m}$	A binary variable which equal to 1 if the r^{th} potential run of machine m has been assigned to an operation, 0 otherwise;
ω	weighting parameter in whited sum method
F^{trans}	Transformed i^{th} term of objective function
F_i^{max}	The upper limits for transformed i^{th} term of objective function
F_i^{min}	The lower limits for transformed i^{th} term of objective function

Part I

Distributed Layout Design

Chapter 1

Introduction

1.1. Facility Layout Design

History of structures goes back to thousands of years, for example, the Egyptian pyramids and Harappa and Mohenjo-Daro civilizations of the Induce Valley, suggest that the layout problem has been considered by facility designers for thousands of years. The industrial revolution presents some examples of the significance of layout in designing factories. The modern assembly line and its basic concept is credited to Henry Ford, who perfected the factory layout to achieve higher level of productivity and efficiency. However, the study of facility layout problems via complicated mathematical models did not began until the introduction of quadratic assignment problem in mid-1950s by (Koopmans and Beckmann, 1957).

It is estimated that roughly 8% of US gross national product has been spent annually since 1955 on new facilities (Tompkins *et al.*, 1996). In addition, expenses for modification of previously purchased facilities are also significantly high. Tompkins *et al.* (1996) claimed that 20-50% of total operating costs are related to material handling cost. Therefore, developing an efficient facility plan not only can reduce these costs to 10-30% but also can increase productivity as

well. More importantly, studies on manufacturing systems indicate that 30-50% of the cost of a product is related to material handling expenses (Sule, 1994). Thus, optimized facility arrangements can reduce product costs and enhance competitive position of companies.

1.2. Basic Type of Layout

In designing manufacturing facility, the first step is to determine the general flow pattern for material, parts, and work-in-process (WIP) inventory through the system (Heragu and Ekren, 2010). Flow pattern refers to the overall pattern in which the product flows from beginning to end. The second step is determining the type of layout to be used. We discuss five types of layouts here.

Functional Layout

The functional layout is also known by other names such as process layout or job shop layout. In a functional layout, machines with similar functionality are grouped into a single department. The departments are located related each other in order to increase machine utilization and production flexibility. A functional layout is shown in Figure 1.1-a to have three types of machines where each geometric shape represents a particular machine type. This type of layout typically is used when product variety is high and/or production volumes are small. The advantages of functional layout are that the utilization rate of each machine station tends to be quite high and there is a tight span of control. However, a functional layout is notorious for its material handling inefficiency. An example of tangible manufacturing firms is semiconductor wafer fabrication where the design of most wafer fabrication facilities has followed functional layouts. To perform several operations, distance approximately traveled by a 300-mm wafer during its 250 process tools visitations is 8-10 mi (Agrawal and Heragu, 2006). It shows

functional layouts can negatively affect material handling efficiency. Scheduling complexity and vulnerability to change in demand and product mix or routing are among other weakness (Benjaafar *et al.*, 2002).

Cellular Layout

In contrast to functional layout in cellular manufacturing, Figure 1.1-b, machines are grouped into cells based on product families, parts similar in size or parts created using similar manufacturing steps. Typically, each cell is dedicated to a single product family. Compared with functional layout, cellular layout includes faster throughput time, less material handling, less work-in-process, and reduced set-up time, but cellular layouts are notorious for their inefficiency when introducing new products and also could be severely affected by changes. Although cellular factories can be quite effective in simplifying workflow and reducing material handling, they can be highly inflexible since they are generally designed with a fixed set of part families in mind (Benjaafar *et al.*, 2002). When the part mix changes typically more machines are required to compensate for improper balancing. Cellular manufacturing also may not be appropriate if operation sequence or routing is prone to change in time.

Product Layout

Product layouts (or also known as flow shop layouts) largely depend on process of products or a product mix. In theses layouts (see Figure 1.1-c), machines are once again laid out in accord with the needs of one product or a small product family. However, they are arranged in a linear layout according to the operation sequences of product or the product family. Product layouts are designed for high volume (mass) production of a single product to justify the use of expensive dedicated machines and equipment (Khaewsukkho, 2008). One disadvantage of flow lines is

the lack of flexibility. They are not able to produce products for which they are not designed. If the design of product is altered, a major reconfiguration of the linear layout may be required. If new products are introduced, it is absolutely essential that the new line with additional investment properly be set up.

Hybrid Layout

Expanding production range and capabilities requires machine variety. In this case, companies find that their existing layout type dose not meet their needs entirely. Hence, they may adopt a cellular layout added to the facility which already has a production-line layout. Figure 1.1-d shows a sample hybrid layout that combined a functional, cellular and product layout.

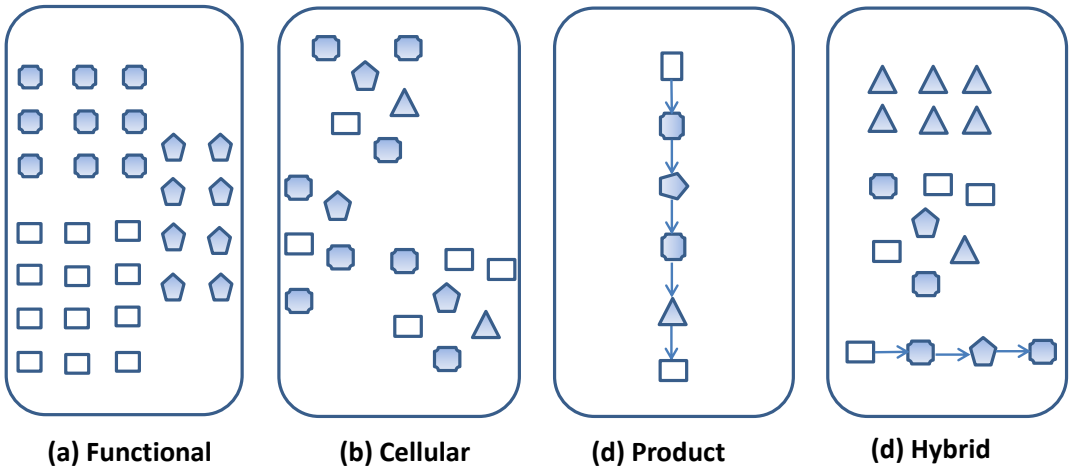


Figure 1.1: Four types of traditinal layouts

1.3. Next-Generation Facility Layouts

In today's dynamic environment, a properly designed facility can bring about competitive advantages when it operates at low cost, responds to the need for

efficient and fast delivery of products, accommodates frequent new products introduction and tackles unexpected demand fluctuations. With regard to environmental uncertainties, manufacturing facilities must be able to cope with both internal changes and external forces. The internal disturbances are considered equipment breakdowns, variable task times, queueing delays, rejects, and rework while external forces refer largely to the fundamental uncertainties of the competitive environment where product demand, design and type are highly volatile. Figure 1.2 shows a case study of an equipment manufacturer for the semi-conductor industry. The example illustrates how todays product demand fluctuates in volatile markets: within five years the annual product demand for a specific machine type changes from 100 machines to 5 (in year 2 and 3) and backs again from 20 to 120 machines. Manufacturing systems that produce multiple

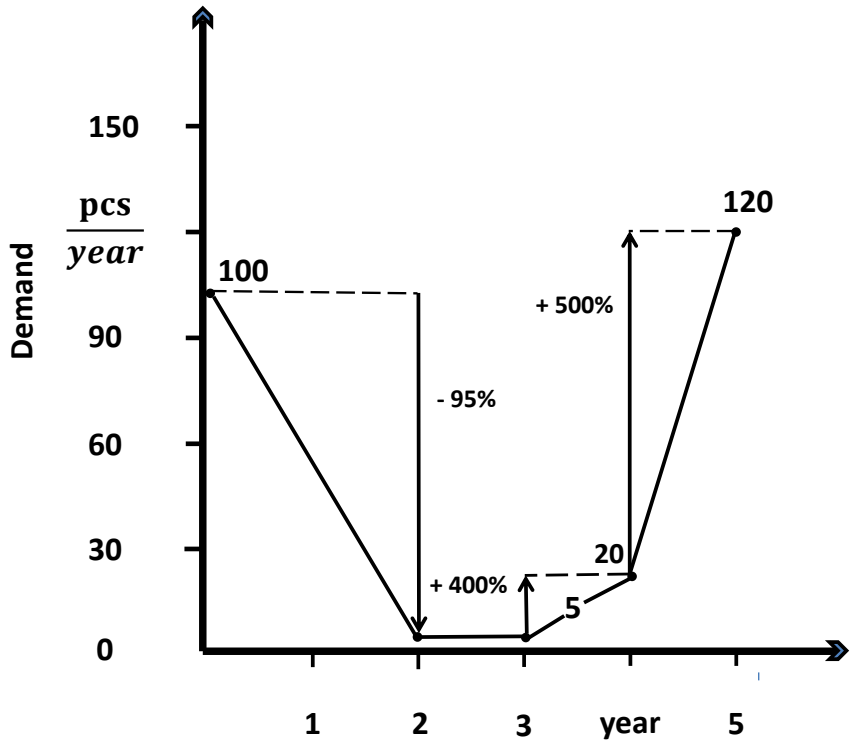


Figure 1.2: A Case study on product demand in turbulent markets (Schnsleben, 2007)

components and function in highly volatile environments are increasingly challenged to meet consistently high levels of operational efficiency and flexibility. This type of manufacturing environment has complex operating characteristics, such as high variety in product mix, uncertainty in production demand, and many different manufacturing routings resulting in a complex material flow network (Khaewsukkho, 2008). This is evident, for example, in the automotive industry, the semiconductor and Electronic Manufacturing Service (EMS) industry. A EMS providers is a contract manufacturer can produces parts, modules, devices and complete system solutions across a wide range of industries, including automotive, aviation, medical devices, industrial and office electronics, measurement, telecommunications and consumer products. Today, EMS providers increasingly act as sole producers of the Original Equipment Manufacturing (OEM)'s finished products, while the OEMs remove themselves from manufacturing entirely (Gentry and Elms, 2009). In contrast to other providers, an EMS firm usually doesn't have its own product line, but rather provides manufacturing services. For instance, Zollner, as one of the world's top 15 EMS companies, performs about $3,000$ manufacturing products per year for about 600 customers. With an acute sense of the industry drivers, Zollner organizes its own manufacturing capacities in a complex pull principle that effectively anticipates batch size demand.

There is an emerging consensus that the traditional layout configurations do not meet the needs of the multi-product enterprise and that there is a need for a new generation of factory layouts that are more flexible, modular, and more easily reconfigurable (Benjaafar *et al.*, 2002). This is mainly because traditional layouts, functional and cellular layouts, are generally developed assuming stable demand and product mix for a considerable long planning horizon. For example, as mentioned previously, in hi-tech industries such as consumer electronics, telecommunications equipment and semiconductors the demand is volatile and

challenging to manage due to the rapid rate of innovation causing short product life cycles. Hence, in order to adapt to such changes firms nowadays have been seeking efficient approaches to design layouts that are more flexible and responsive. Other researchers such as Benjaafar and Sheikhzadeh (2000); Irani and Huang (2000); Kochhar and Heragu (1999) and Montreuil (1999) have also emphasized the need for alternate types of next-generation facility layouts. Modern layouts must lower inventories, decrease production lead times, and offer greater flexibility for product customization, in addition to minimizing the traditional measure such as minimizing material handling trips (Heragu and Ekren, 2010). Numerous articles have been published in design of flexible facility layouts. Examples of these layouts are fractal layouts, virtual cellular layouts, modular layouts, and distributed layouts.

Fractal layouts (Venkatadri (1997) and Askin *et al.* (1999)) divide the facility into smaller cells or fractals. Each fractal is identical and can produce a wide variety of products. These are considered as small factories within a factory. In this type of facility layout design, material handling distances are only optimized within each fractal. Manufacturing system using fractal layout can benefit from the decentralization. Although fractal layout is an extension of a more generic approach to cellular layout, there is difference between cell and fractal because there is no specialization to produce certain parts or family of parts in a fractal. Hence, they are more flexible to demand and product mix change. The fractal can be formed based on machine type such that at least one machine of each type is included in a cell, Figure 1.3-b. Therefore, the machine type with the minimum number of machines determines the number of fractals. Additional machines of a type are allocated as evenly as possible. That may allow flow performance optimization and avoid unnecessary duplications compared to identical fractal layouts, Figure 1.3-a.

Drolet (1989) discussed *virtual cellular layouts* when a job order needs a set

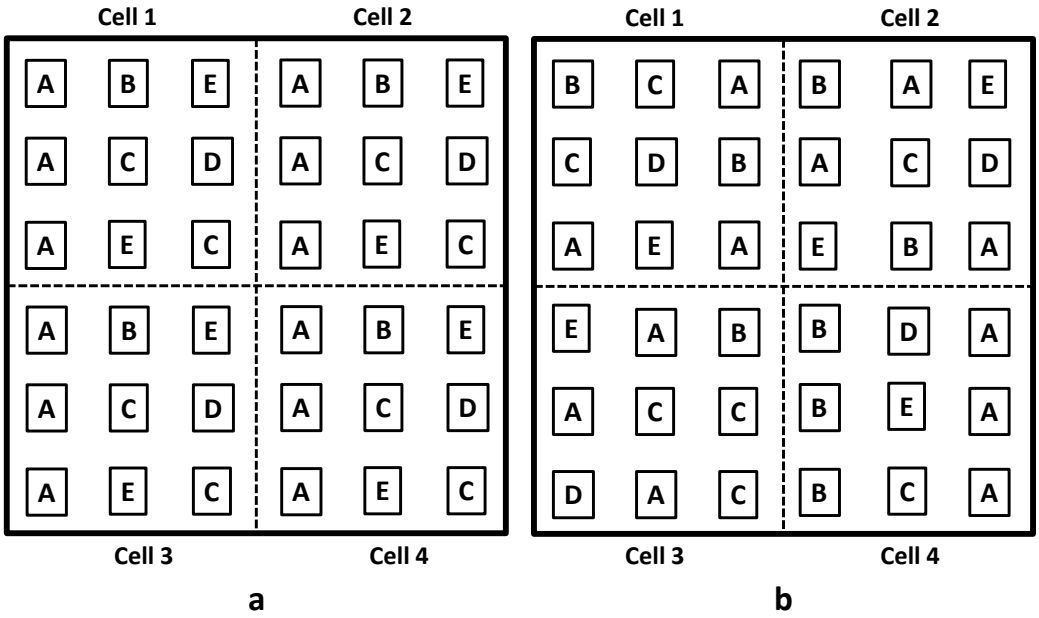


Figure 1.3: Fractal layouts with (a) identical and (b) non-identical configurations

of machines to be put together, a quick temporary cell consisting of adjoining machines are formed. A machine is either a member of a pool of available machines or member of a virtual cell. Although this concept may seem not much different from a typical cellular approach, it sheds some light on the concept of moving replicates of individual machine types to nonadjacent locations (between cells) to satisfy customer order requirements (Ganesan, 2007). Figure 1.4 shows a system that has three active virtual cells where two of the cells shared a workstation.

Modular Layouts (Irani and Huang, 2000) can be constructed as network of basic modules. Layout modules can be categorized as flow line, cell and functional modules. As the product mix and demand changes, certain layout modules will be eliminated and others added. By doing this, the complex material flow network in a multi-product manufacturing facility is addressed. The proposed concept uses the idea of grouping and arranging the machines required for subsets of operations in different routings into a specific (traditional) layout configuration

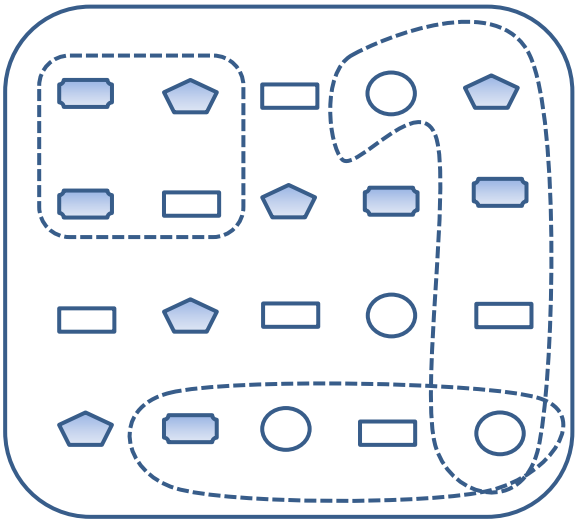


Figure 1.4: Virtual Cellular Layout

that minimizes total flow distances or costs (Benjaafar *et al.*, 2002). A planer needs to determine number of layout modules, to which module in the existing layout each machine is allocated and to which machine each operation of each product is assigned. Since layout modules are developed based on an analysis of operation sequences, this approach to facility layout would allow the facilities planner to customize the layout for any facility based on the unique composition of the product mix processed in that facility (Irani and Huang, 2000).

The metric for evaluating the efficiency of a layout type has also changed in today's factories. Robustness and flexibility or reconfigurability are of importance. Robust layouts can behave well over multiple production periods and different scenarios that are suitable where both uncertainty of future production requirements and the cost of re-layout are high. Robustness is typically achieved by duplicating key processes or machines at strategic locations within the production facilities. Flexible layouts can be reconfigured with minimal effort to meet the changes, such as reconfigurable layout where there is a high level of

uncertainty but the cost of re-layout is low. It aligns itself with the notion of real-time enterprise in which the changes to layout context are readily available, and it keeps operating on the edge by doing real-time layout adjustment with live data (Meng *et al.*, 2004). It has the advantage of minimizing the material handling cost by reconfiguring a layout when warned by changes. Of course, this cost must not be more than the cost of relocating shop-floor equipment.

1.4. Distributed Layouts

In world-class manufacturing systems, machines are generally more versatile. Machining centers that perform milling, drilling and boring operations are already present in today's factories. The end result of machine versatility is that the products don't need to visit as many workstations as in traditional layout. Many more operations will be performed at every machine. Also, products tend to have more natural, simpler design as the result of components standardization and sustaining effort throughout the simplification an ingenuity of process automation. Given these, it is clear that versatile machines need to have to be close to each other when dealing with factory of future (Drolet, 1989).

The concept of distributed layout (DL) expands current thinking of methods for facility layout, and supports the need for a new generation of facility layouts beyond the traditional layouts that continue to be studied and implemented in industry. In distributed layouts, similar departments (machines) are distributed throughout the factory floor to increase the accessibility to these resources from the different regions of layout (Baykasoglu, 2003). As a result, efficient flows could be more easily found for a larger set of product routings, which would then tend to diminish the need for rearranging the layout even when production requirements change significantly (Lahmer and Benjaafar, 2005). To design this

layout, the planer should decide how to disaggregate and duplicate the subdepartments or machines throughout the plant (Montreuil and Venkatadri, 1991). Figure 1.5 represents the schematic structure of layout with varying degree of distribution. In the fully distributed (holonic) approach, machines are spaced randomly as evenly as possible throughout the entire facility. For functional partially distributed layouts, facility designer restricts all (some) of the machines of the same type to be in adjacent locations and requires that these aggregated machines have reasonably compact shapes (Lahmer and Benjaafar, 2005). Similarly, the functional layout is also a constrained version of the distributed one. A holonic layout might appear chaotic, but it is the ability of the system to provide many alternatives that allows it to adapt to changing conditions (Askin *et al.*, 1999).

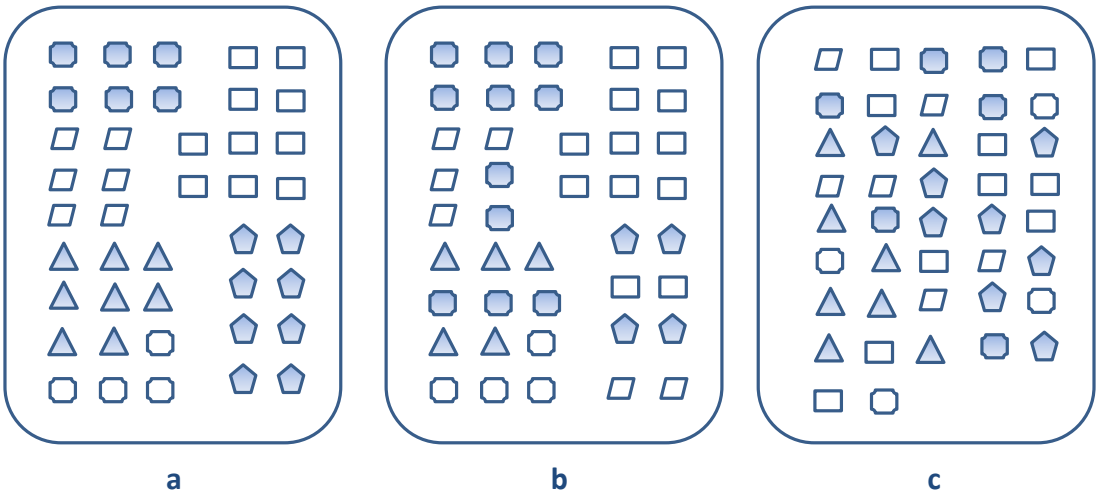


Figure 1.5: 1.Layouts with varying degrees of distribution: (a) functional layout, (b) partially distributed layout, and (c) maximally distributed layout.

1.5. Layout Design and Analysis

According to Meng *et al.* (2004), to put a layout problem into context, the entities and activities relevant to the layout problem can be organized into a layered model as shown in Figure 1.6. The reference model has three physical layers: product mix, machine types and locations on the shop floor. Product mix includes the types of products that need to be produced and their arrival volumes (parameters related to demands). Each available machine belongs to a machine type. The number of locations on the shop floor is equal to the total number of machines. The two logical layers of the reference model denote the design activities involved. The process planning problem maps each product to a sequence of machine types, its output (product routings) includes other production data such as processing time, setup time and tooling information. The layout problem is to find a one-to-one mapping from machines on the machine type layer to locations on the shop floor. The third logical layer (not shown) pertains to the scheduling problem. It finalizes the machine types in a product routing to specific machines on specific shop floor locations, and coordinates the timing, sequencing and prioritizing of all work orders assigned to one machine. The solution of the layout problem is determined by entities and activities in other layers of the reference model. Define the collection of these entities and activities, i.e. product mix, product routings, machines and locations on shop floor) as the context of a layout problem. As long as the context is fixed, theoretically speaking, there exists an optimum layout for this context.

By definition, the facility layout problem is a simple assignment of m machines to n locations on the shop floor. What makes the facility layout problem difficult to solve is the large combinatorial search space, especially when n is large (possibly $n!$ feasible solutions if no special restrictions on the locations of specific machines or relative location of a subset of machines), and the construction of a score function which incorporates various business considerations to evaluate the

goodness of a layout.

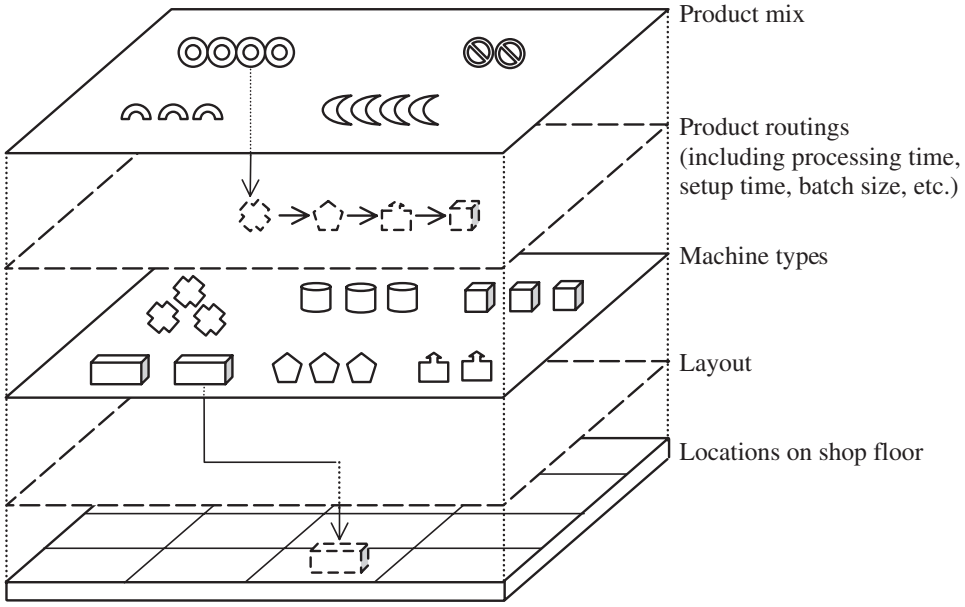


Figure 1.6: A Facility layout problem model (Meng *et al.*, 2004)

.

Dynamic System Reconfiguration

Another aspect of facility design prescribed to address the challenges of meeting high operational efficiency and flexibility in highly volatile environments is dynamic system reconfiguration (DSR). In the dynamic approach, the layout plan is based on a multi-period time horizon. During this time if the material flow changes warrant it, layout rearrangements may be planned in one or more periods. The analysis is based on the trade-offs between the costs of excess material handling if a layout is not rearranged when required and the costs of such rearrangements. With emerging technology to support reconfiguration, the objective of layout design is shifting from long-term material-handling efficiency to short-term responsiveness (Benjaafar *et al.*, 2002). Management can focus on

operational performance by reconfiguring layouts more frequently to relieve short-term congestions and maximize throughput for current product mix and demand. Thus, the nature of decision on layout reconfiguration is becoming more tactical, rather than strategic. Altering an existing layout introduces two kinds of costs: (1) the cost incurred due to loss in production time, and (2) the cost of physically moving equipment from their existing location to the new location. This includes planning, dismantling, construction, movement and installation costs (Kochhar and Heragu, 1999). However, technological advances have been reported to enable DSR. Southwestern Industries (www.southwesternindustries.com) developed a compact and mobile milling machine (TRAK QuikCell QCM-1). The machine is small enough to fit through most doors, and its rigid frame does not require re-leveling after each move. There is also a shift to lighter machine tools driven by advances in materials and processing technologies (Heragu and Kochhar, 1994). Systems that allow easy storage and retrieval of large equipment and machine tools have also been developed. For example, Robotic Parking (www.roboticparking.com) developed a modular automated parking system. Although originally designed for car parking garages, the technology is finding applications in manufacturing where machine tools can be stored and retrieved as needed (Benjaafar *et al.*, 2002).

Comprehensive Models

The primary design strategy for design of a layout is how and where to allocate machines such that material handling cost can be minimized. However, during the placement and floor-planning phase, several important design challenges have to be simultaneously addressed in order to maximize the benefits of layouts design. The facility layout, material handling system, process routings and production plan must all fit together to enable a competitive manufacturing performance (Askin *et al.*, 1999). In respect to the design of facility layout, considering several

aspects of a manufacturing system can practically improve layout's effectiveness and functionality. As pointed out in Arvindh and Irani (1994), an integrated approach should be pursued in manufacturing system analysis, since different aspects of a system are interrelated in many ways. In addition, a comprehensive model consisting of different attributes of a system can help one to understand the problem better. Integrated system approach can minimize the possibility of certain important aspects of a system being overlooked, while other issues are being studied. Review of recently published articles in the facility layout design showed that these factors are associated with lot splitting, sequence of operations, alternate part routings, operation time and cost, cost of subcontracting part processing, machine capacity, setup cost, tool consumption, workload balancing, and machines separation constraints.

Given the new reality that layouts are likely to change very frequently, possibly every few months rather than years and that, at best, we only have knowledge of production activities during the upcoming planning period, we need to develop a layout only for the next planning period. In addition, due to the short term life of given layout and availability of production data for this time period, it is possible to consider optimizing operational performance measure such as WIP inventory, setup and part cycle time (Heragu and Ekren, 2010). Thus, the nature of decision on layout reconfiguration is becoming more tactical, rather than strategic. This makes the integration of dynamic system reconfiguration (DSR) with tactical decisions such as production planning a sensible approach. According to Defersha and Chen (2009a), the majority of literature in DSR assumed that the production quantity in each period is equal to the demand of the same period. In reality, however, production quantity can be different from the demand as it may be satisfied from inventory or by subcontracting. Thus, the production quantity should be known in order to determine the number and types of machines to be installed in the system. However, the number and types of machines to be

installed should in turn be known first in order to determine production quantity due to capacity considerations. This implies that dynamic system reconfiguration and production planning problems should be solved simultaneously in order to obtain the overall optimal solution. Here it is important to note that it is relatively easy to get detailed data on material flows, machine setup, processing time and other relevant information relative to production activities in the next period as opposed to manufacturing activities for the next five years.

In summary, the ability to design and operate manufacturing facilities in factories that must deal with high product variety or high volatility in their production requirements is becoming increasingly important. Such challenges can be addressed by using distributed layouts in such manufacturing system (Krishna *et al.*, 2009; Lahmer and Benjaafar, 2005; Benjaafar and Sheikhzadeh, 2000). In addition, distributed layouts in settings with multiple periods (Dynamic distributed layout) can take the advantage of the DSR to enhance its robustness to uncertainty and variability (Lahmer and Benjaafar, 2005). Moreover, taking account of a comprehensive distributed layout design model simultaneously considering for a variety of manufacturing attributes can provide a framework within which aspects of a manufacturing system can be integrated to increase layout's effectiveness and functionality.

1.6. Aim and Objective

The primary objectives in our research are two-fold. The first objective is to develop a comprehensive mathematical model to design a dynamic distributed layout. The ideal integrated layout design model would integrate all design factors such as machine allocation to pre-specified location, flow allocation to overlapping process routes, production planning and inventory control and some other attributes that capture several aspects of manufacturing system in reality. The

comprehensive model similar to the one proposed in this study usually imposes computational difficulties and for only small size problems may be solvable using off-the-shelf optimization packages due to NP-complexity. For solving such difficult problems, two hybrids metaheuristics (including Linear Programming Embedded Simulated Annealing (LPSA) and Linear Programming Embedded Genetic Algorithm (LPGA)) and a metahuristec (simulated annealing) algorithm are developed, and the results obtained are compared. The proposed formulations and solution procedures make an important step towards the development of distributed facility layout models. These models and solution procedures can be applied to a variety of manufacturing settings with minor modifications. The second objective in our research is to provide a scheduling algorithm for controlling manufacturing systems with distributed layouts. To our knowledge, a model for scheduling in distributed layout manufacturing system has not been addressed in the literature.

Chapter 2

Literature Review: Facility Design

2.1. Introduction

In this review, we present some of the previous work on distributed facility layout design problem. As the literature on facility layouts design is enormous, we provide a limited, brief summery in this topic. Our goal is to provide references on the studies which related parts of the design of distributed layouts addressed in this thesis. Other reviews of interest are Drira *et al.* (2007) for facility layout problems, Meller and Gau (1996) for block layout design, Shahbazi *et al.* (2013) for Quadratic Assignment Problem, Blum *et al.* (2006) for hybrid metaheuristic and Balakrishnan and Cheng (2007) for dynamic facility layout. The organization of this Chapter is as follow. In Section 2.2, we review block layout design and discuss traditional facility layout design problems. Sections 2.3 and 2.4 are dedicated to modern facility layout design and presents the literature on distributed facility layout design. Section 2.5 reviews the solution procedures used to solve distributed layout problems in the literature.

2.2. Traditional Facility Layout Problem

The traditional facility layout problem is associated with assigning m departments to n distinct locations in different areas of floor plan. As defined in the literature, minimizing the material handling costs inside a facility is usually the only objective of the problem. The objective function used to evaluate the goodness of a layout is subject to two sets of constraints: (1) department and floor area requirements, and (2) department locational restrictions (departments cannot overlap, must be placed within the facility, and some must be fixed to a location or cannot be placed in specific regions) (Meller and Gau, 1996). The center of much of the research in the traditional layout literature has been on the efficient solution of a combinatorial optimization problem which has resulted in a block layout (see Figure 2.1-a). In the following step, detailed layout plan associating with exact department locations, aisle structures, input/output (I/O) point locations, and the layout within each department are created (see Figure 2.1-b) (Meng *et al.*, 2004).

Mathematical programming approaches such as nonlinear and mixed integer programming have been used to solve the traditional layout problem. However, many researchers turned away from mathematical programming technique to the metaheuristics such as genetic algorithm and simulated annealing due to lack of solution quality and computational cost in mathematical programming techniques. These stochastic search techniques have been discovered as a useful tool for a wide range of combinatorial optimization problems. A typical formulation of the traditional facility layout problem is the quadratic assignment problem (QAP) which is introduced by Koopmans and Beckmann (1957). In this formulation, layout is considered as discrete as shown in Figure 2.1-a. Discreet representation is not suitable to map the exact position of facilities compared to continual representation, Figure 2.1-b, that appropriately is able to consider the orientation of facilities. The QAP formulation of layout design is presented in

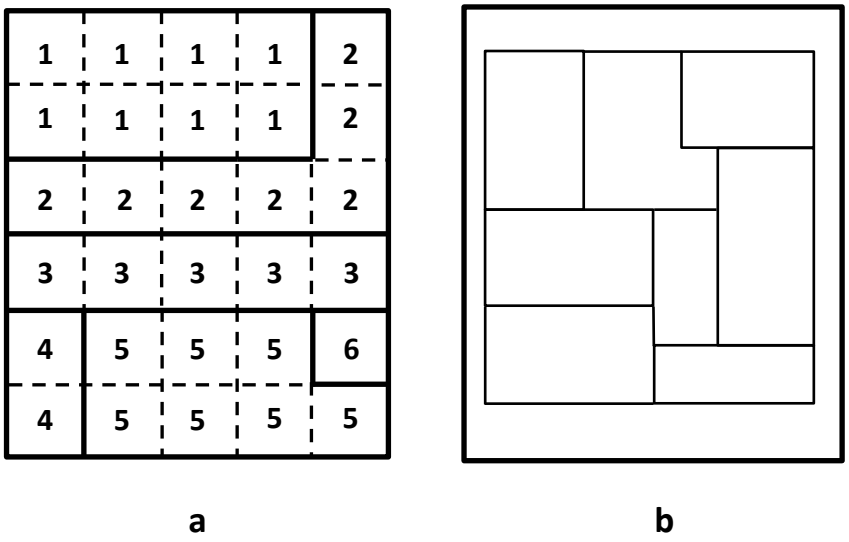


Figure 2.1: Discrete (a) and Continual layout Representation (b)

Eq. (2.1) to Eq. (2.4). The material handling and facility installation costs are minimized using the objective function. Most authors ignore the second term in the objective function because it is linear. The cost of assigning a department at a particular location is dependent on the location of the interacting departments. Such dependency leads to the quadratic objective that inspires the problem's name. The constraints in Eqs. (2.2) and (2.3) are to enforce that each location must be occupied by only one facility and each facility must be assigned in only one location. One of the disadvantages of the QAP is that it requires all departments be of identical shape and size (equal-area) and the location of site (the physical area to be occupied by a department) are known priori. Therefore, this approach is incapable of solving layout design problems when the location sites are not specified in advance. It also has been shown as a NP-hard problem which implies that, in general, it is a hard problem to solve (Sahni and Gonzalez, 1976).

$$Min \sum_{l'=1}^{m}\sum_{l=1}^{m}\sum_{j=1}^{m}\sum_{i=1}^{m} d_{ll'} \cdot f_{ij} \cdot y_{iljl'} + \sum_{l=1}^{m}\sum_{i=1}^{m} c_{il} \cdot x_{ip} \tag{2.1}$$

$$\sum_{i=1}^{m} x_{il} = 1 \quad \forall i \tag{2.2}$$

$$\sum_{l=1}^{m} x_{il} = 1 \quad \forall l \tag{2.3}$$

$$x_{il} \in \{i, l\} \quad \forall i, l \tag{2.4}$$

m Number of facilities (locations).

c_{ik} The cost of assignment of facility i to locations l.

$d_{ll'}$ Distance between locations l and l'.

f_{ij} Material flow from facility i to j.

x_{il} A binary variable equal to 1 if facility i is located at location l; 0 otherwise.

Kusiak and Heragu (1987) and others consider the unequal-area facility layout problem as modified QAP such that they break the departments into small girds with equal area and assign a large artificial flow between those girds of the same department, to enforce that they are not disaggregated, and solve the resulting QAP. However in this approach, since the number of departments is increased the problem is not easily solved even for small size problems. Detailed survey on QAP and its variants can be found in (Shahbazi *et al.*, 2013).

Mixed Integer Programming Problems

The unfavorable computational time and poor solution quality obtained from QAP-type models motivate the researchers to develop mixed integer programming models to address a continual representation of the facility layout design where

all the facilities are placed anywhere within the planar site and must not overlap each other (Das, 1993; Dunker *et al.*, 2005; Meller *et al.*, 1998). The mixed integer programming approach is powerful to take into account specific information of the problem and determine a comprehensive result about the system, however, it is not capable to achieve optimal solution for even small size problems. Several researchers have developed heuristic and metaheuristic-based techniques to solve realistic-sized instances of the problem. A review of facility layout design using Mixed-integer is performed by (Drira *et al.*, 2007).

Limitation of Traditional Facility Layout Design

The design of facility layout considers a number of issues including: (1) the determination of locations where departments will be established, (2) overall layout establishment, and (3) detailed layout plan development such as the location of each machine, and the determination of material handling methods to be used. To achieve an efficient layout, these issues should be examined simultaneously. However, traditional facility layout problems are generally formulated and solved sequentially due to the complicated nature of the integrated problem. For example in cellular layouts, the traditional facilities layout problem considers the area of cells, but not the shape, and has been modeled in the past in several ways, such as a quadratic assignment problem (QAP), a quadratic set covering problem (QSCP), a linear integer programming problem, and a graph theoretic problem (Kusiak and Heragu, 1987). A small change in one department can affect the entire system layout, and a different layout may require different forms of departments for achieving the best system efficiency. Another limitation in addressing traditional layout problems is the design of layout for a single planning period where the potential to frequently alter layouts is ignored.

2.3. Modern Facility Layout Design

In the modern facility layout design, it is common that some design factors such as machines dimensions, capacity of machines, production volumes, processing routes, etc., are to be considered together to achieve a good facility layout in a manufacturing environment. For example, this occurs in cellular manufacturing systems design, where cell formation problem, machines allocation within each cell and the sequencing of operations are addressed simultaneously instead of formulating and solving these problems sequentially (Gupta *et al.*, 1996). The accelerated use of cells in industry in the 1990's caused that the cell formation problem commanded wide popularity among researchers. While much of the early research has considered only the part-machine incidence matrix to form the cells, recent studies have incorporated additional information such as the operation sequence, demand volume, processing time and capacity considerations (see Park and Wemmerlöv (1995) for a review of these studies). Park and Wemmerlöv (1995) presents a taxonomy based on input data categories. They suggest that cell formation problems be classified based on the type of data they need to produce solutions. The basic types of input data are: (1) Part machine type requirements data (PMR), (2) Sequence data (SEQ), (3) Volume data (VOL), (4) Processing time data (PRT), and (5) Capacity data (CAP). An example of input data-based classes are given in Table 2.1. According to the authors, higher class

Table 2.1: An example of input data-based classification techniques

Class	Input Data Types
I	PMR
II	PMR+SEQ;PMR+VOL
III	PMR+SEQ+VOL;PMR+VOL+PRT
IV	PMR+VOL+PRT+CAP+SEQ
V	PMR+VOL+PRT+SEQ;PMR+VOL+PRT+SEQ+CAP

techniques, by relying on more types of input data and considering more objective

/constraints, solve more sophisticated cell formation problems and appear more likely to generate solutions that can achieve objectives specified by the user. In other words, techniques in classes IV or V have higher face or model validity than techniques in classes I or II. Situations may exist where a user prefers a lower class technique (due to availability, ease of solution generation, etc.) to derive an incomplete solution, i.e. not all decision variables of concern to the user are included in the model structure, and then use heuristic approaches (external to the CFT) to finalize the solution. However, the result obtained by Arvindh and Irani (1994) illustrates the need for solving simultaneously all subproblems in cell design problem instead of solving them in non-sequential manner. This is essential because of the highly interacting natures of subproblems.

There are also limitations underlying many of the traditional methods used to design and evaluate factory layouts making them less effective in factories with high variety in product mix, uncertainty in production demand, and many different manufacturing routings resulting in a complex material flow network. With routing information embedded in the layout, the design of layout configurations becomes possible such that the obtained layout is most robust over that range of uncertainty (Benjaafar and Sheikhzadeh, 2000).

More importantly, some studies suggest that enforcement of facilities to get arranged in a pre-specified layout shape may increase the total distance traveled by the materials. For example, Urban *et al.* (2000) consider a model that do not require the machines to be placed in a functional layout or in a cellular arrangement, but allowed the material flow requirements to dictate the machine placement. However, the model formulated in Urban *et al.* (2000) supposes that the location of sites is known a priori. While limiting the layout design to a particular shape has advantages (e.g., simplified workflow and routing), they usually lead to suboptimal layout designs in terms of total distance traveled (Benjaafar and Sheikhzadeh, 2000).

Dynamic Facility Layout Planning

Traditionally, the facility layout design is usually considered as static. This means that once the facility layout is designed and executed, it remains unchanged for years. This is not the case in today's dynamic environment for manufacturing firms where the layout may be reconfigured as frequent as every few months. Due to pressures on manufacturing systems to adapt to change, the idea of dynamic layout problems has been introduced by several researchers. Rosenblatt and Kropp (1992) indicate that if the objective is to minimize the expected material handling cost, then it is equivalent to solve the layout problem that results from the expected flow matrix. Such a result allows one to effectively use traditional layout algorithms that consider one flow matrix (Meller and Gau, 1996). This model takes into consideration material handling cost as well as cost of relocating machines from one period to the next. A majority of the research that has been done in the dynamic facility layout planning (DFLP) assumes equal department sizes and deterministic material flow. A review of some previous work on dynamic layouts is presented in (Balakrishnan and Cheng, 1998). Palekar *et al.* (1992) consider uncertainties explicitly in plant layout design. They consider a stochastic dynamic layout problem assuming the following are known a priori: (i) material flows matrices for each of several planning periods, and (ii) the transition probability from one flow matrix to another. A restriction of the DFLP is that it necessitates production data for multiple periods be available at the initial design stage. This requirement is increasingly difficult to fulfill in today's environment, where factories are suffering by the unavailability of production data for more than one period at a time (Benjaafar *et al.*, 2002). Another promising approach to address changes in the production environment is to build inherent features into the layout that enable it to adapt to volatilities. Modular, virtual cell and fractal and distributed layouts are among capable ones addressing uncertainties.

2.4. An Overview of Distributed Layouts Design

Before reviewing distributed layout design literature, some terminologies which are used throughout this thesis are presented in this paragraph. They are static against dynamic production requirements and deterministic versus stochastic production requirements. A static production requirement refers to a single period when designing a DL. That is, product mix and demand for the whole planning period are constant. However, product mix and demand in such cases can be deterministic or stochastic. For static and deterministic production requirements, there is only one possible set of product mixes and demands which are known. In contrast, static and stochastic production requirements have a set of possible product mixes and demands to occur; each has its probability of occurrence. Therefore, designing a DL for such requirements needs to consider all possible product mixes and demands.

As previously mentioned, the distributed layouts are facility layout designs achieved by jointly determining the machine location decisions and part flow allocation decisions given production requirement, the volume and route information (Benjaafar and Sheikhzadeh, 2000; Benjaafar *et al.*, 2002; Urban *et al.*, 2000; Taghavi and Murat, 2011). In particular, the machine placement decisions are made based on the manufacturing requirements as opposed to dictating a particular layout form (product, process, or hybrid) a priori (Urban *et al.*, 2000). A significant amount of literature exists in the area of layout design problem. However; there is limited research in the body of literature dealing with distributed facility layout problem. A list of assumption about the proposed models in sample articles and in this study is given in Table 2.2. The integrated machine allocation and layout problem (ILAMP) is the core part of models addressing uncertainties

in design of distributed problems. The unique aspect of problems in the literature is that the work flow between any two given machines is no longer fixed as with the typical machine layout problem (MPL), it is now a decision variable. Therefore, the objective function is polynomial, instead of quadratic which is in the QAP as discussed before.

Montreuil *et al.* (1993) introduce a holographic or maximally distributed layout where functional department are fully disjointed into individual machines to be strategically placed as far from each other as possible throughout the shop in order to obtain a robust layout for changing product mix and volumes. Using proximity of distinct machines, the intent is to provide efficient process routes for any part type which the system may be asked to produce with minimum notice. As orders arrive, part routings are constructed based on compatibility between part requirements, machine locations and machine availability (Askin *et al.*, 1999). Marcotte *et al.* (1998) extended the concept of holographic layouts and proposed an approach to determine the layouts assuming that the expected number of trips from and to each machine is known. These would be derived from the process plans and demands.

Urban *et al.* (2000) presented a deterministic ILAMP to deal with the uncertainty of product mix for a single period. He suggested that the best way to handle the uncertainty in the product mix is to allow the material flow requirements to dictate the machine placement. Given: (1) a set of product mixes, (2) machines or operations required to produce each product, (3) the time required to process each product on a given machine, (4) the capacity of each machine, and (5) the processing route of each product , the design objective here is to minimize the total distance or material handling cost. The algorithm seeks to create robust layout and determine parts routings simultaneously. They considered equal-sized machines with replicas and assumed that the location of sites is known a priori. The resulting model is aggregated quadratic assignment problem

Table 2.2: A list of assumption about the proposed models in sample articles and in this study

Article	Machine		Layout Presentation		Production Requirement		Layout Evolution	
	Equal-size	Unequal-size	Discrete	Continual	Deterministic	Stocastic	Static	Dynamic
This study	†		†		†			†
Montreuil *et al.* (1993)	†		†				†	
Marcotte *et al.* (1998)	†		†		†		†	
Urban *et al.* (2000)	†		†		†		†	
Benjaafar and Sheikhzadeh (2000)	†		†			†	†	
Lahmer and Benjaafar (2005)	†		†		†			†
(Solimanpur and Jafari, 2008)		†		†	†		†	
Taghavi and Murat (2011)		†		†	†		†	
Baykasoglu (2003)	†		†				†	

which is then solved using two metaheuristics. Their model is closer to reality by considering capacities of machines. Some observations in their publication are worth considering, they reported that by integrating the machine layout and product assignment problems and allowing the material flow dictate the placement of machines, the work flow can be reduced to nearly 29% compared to the functional layout.

Benjaafar and Sheikhzadeh (2000) considered the stochastic version of IMALP and designed robust distributed layouts using demand distribution information. They proposed an iterative method after decomposing problem into layout assignment and flow allocation sub-problems and compared the performance of distributed layouts with functional, random, and maximally distributed layouts. They concluded that cost savings are also significant even in the absence of variability. In the first step of their iterative technique, they determined, for each possible demand scenario, the amount of material flow due to product p between each pair of departments for all the distribution of demand scenarios, the product process routings, and the product unit transfer loads. Then, using the nonlinear mixed-integer programming model, the corresponding optimal layout and the corresponding optimal flow allocation between copies of the same department were generated. Note that flow matrices generated in first step only give the amount of flow between department types. The determination of the flow volumes between individual departments was determined by the optimization model, simultaneously with the layout. The ILMP model usually assumes workload assignment to machines is possible as long as capacity constraints are not violated. Such a total flexibility, while allowing us to minimize material handling costs, may result in unbalanced workload distribution among machines of the same type. In turn, this may grow the congestion levels and increase throughput times at the more utilized machines. The authors also addressed work load balancing to balance workload assignment among all departments of the same type. Like Marcotte *et al.* (1998),

the authors considered equal-sized machines with replicas and assumed that the location of sites is known a priori but this may not always hold in practice. When an uncertain environment in which the exact values of the probabilities of the different possible scenarios are unknown, layout flexibility can be defined in terms of the robustness of the layout's performance under different scenarios. Thus, the most robust layout is the one whose cost performance remains close to the optimal layout for the largest number of scenarios. Benjaafar and Sheikhzadeh (2000) included a robustness constraint that ensures the material handling costs under any given scenario to be within a specified range of the optimal layout for that scenario. They also reported if the distribution of flow patterns is predetermined, then including flow information at the design stage can lead to better quality layouts. However, material handling costs can be significantly reduced even if no flow information is included in the model (e.g. by a random distribution of departments). In addition, they mentioned that distributed layouts are quite insensitive to inaccuracies in the demand distribution.

The result obtained from Benjaafar and Sheikhzadeh (2000) motivated Baykasoglu (2003) to design distributed layouts without demand or part variety data. However, he considered distributed layouts where the emphasis is on the distribution of resource elements instead of machine duplications. Accordingly, the objective of capability-based distributed layout approach is defined as minimizing total distance of accessing every capability from every location of the layout. His results showed that the proposed distributed layout approach outperforms functional layouts in implementing virtual cells, which was the second motivation of the research. The formation and scheduling process of virtual cells were clearly explained and researched in detail in the literature (Montreuil *et al.*, 1991; Benajaafar, 1995; Marcotte *et al.*, 1998). However, the layout issue had not been addressed entirely since a model for the implementation of a distributed

layout prior to forming virtual cells was developed by (Baykasoglu, 2003). Author supposed all machines are equal-size, but the number of machines is not equal to the number of girds. This machine assignment problem was solved such that the summation of the distances from every location that is unoccupied by a capability to the location where the capability is available is minimized. As a major assumption machine capabilities were defined in terms of resource elements (REs). REs can effectively define the unique and overlapping capabilities of machine tools (Gindy *et al.*, 1996). A brief explanation of REs is given in the next Section because we also widely used the concept of the RE in this study. According to author of Baykasoglu (2003), designing distributed layouts through considering machining capabilities is more beneficial and therefore the capability-based distributed layout should be preferred to a machine-based one, especially in the absence of demand data.

Lahmer and Benjaafar (2005) extended stochastic single period model in Benjaafar and Sheikhzadeh (2000) to dynamic distributed layout model. One of the key features in their design procedure is that the functional departments are disaggregated into sub-departments and are then distributed throughout the plant to form a distributed layout. In this perspective, the functional layout readily is a constrained version of the distributed layout. They assumed: (1) demand information (2) production requirements (i.e., number of products, demand for each product, process sequences and processing times, and department capacities) for each period is known at the initial design stage, and (3) department disaggregation level is also a given factor. The above information is used to develop a mathematical model to determine the locations of each department duplicate (machines) and volume of flow between them in each period. Their formulation also selects part routings, its processing requirement at each department copy and each demand for each period. They also explicitly modeled the production capacity of each department, which can vary from copy to copy and from period

to period. Like most works in design of distributed layout, their model assumed equal-size department duplicates. However, they relax the assumption that assumes the number of locations and departments duplicates are the same by using dummy departments with zero flows and zero rearrangement costs. They also added three terms in the objective function allowing some practical issues to be included in their model; these are associated with workload balancing within duplicates, checking for layout robustness, and discouraging splitting of flows among multiple copies. It is worth to mention that flow allocation is given highest priority over other factors by most of researchers in designing of a distributed layout. However, it needs to be carefully addressed to restrict full flexibility in assigning workload among duplicates of the same department (or same machines). Lahmer and Benjaafar (2005) suggested that it is sensible approach when there is ample processing capacity at each duplicate, the order splitting can be discouraged by restricting the assignment of each product to only one copy. The main contributions of this work are threefold: first firms could benefit from desegregating and distributing functional departments , second the distributed layouts are particularly more beneficial when demand and routing variability are high and product variety is low and third partially distributed layouts outperform the fully disaggregated functional layouts.

Castillo and Peters (2003) extended Urban *et al.* (2000)'s IMALP model to a formulation considered grouping machines to form departments where machines are not necessarily same. Their non-linear mixed-integer model concurrently captured machine assignments, flow allocation and department formation. In contrast to traditional facility layout approaches, the number, shape, and formation of departments are not imposed a priori on the final layout solution and are part of an extended assignment problem. They showed that their proposed facility layout resulted in total material handling cost reduction with an average improvement of over 40% when compared to functional layouts.

There is limited work in the literature to address the problem on the basis of unequal machines and continual presentation of the distributed layouts. Solimanpur and Jafari (2008) and Taghavi and Murat (2011) studied a variation of distributed layout problem proposed in (Urban *et al.*, 2000). A major assumption in their publications is that they consider unequal-area machines duplicates. Accordingly, they developed a mixed integer programming model to address a continual presentation of the distributed layout design where all the facilities are placed anywhere within the planar site and must not overlap each other. Solimanpur and Jafari (2008) showed distributed layouts remarkably decrease the material handling distance compared to an optimum process layout configuration for different problem sizes. The majority of the literature on distributed layout problems to date assumes the flow of material is certain and exact, which is a doubtful assumption in today's dynamic environment. Hosseini Nasab (2014) considered demand information as fuzzy numbers with different membership functions. He believes the assumption is more sensible in today's highly volatile environments compared to stochastic demand. He argued the fuzzy concept is better suited to capture the ambiguity of the demand data due to lack of data in practical cases

2.5. Solution Procedures

As mentioned previously, distributed facility layout problem is special case of the QAP and since the QAP is known to be NP-complete (Sahni and Gonzalez, 1976) it is concluded that the problem is also NP-complete. Thus, heuristic or metaheuristic algorithms are required to develop and solve the problem.

One common approach to solve the problem in the literature is to take advantage of the special structure of the IMALP by decomposing the problem into two sub-problems (i) a facility layout problem, and (ii) a flow allocation problem (Urban *et al.*, 2000; Benjaafar and Sheikhzadeh, 2000; Lahmer and Benjaafar,

2005; Castillo and Peters, 2003; Taghavi and Murat, 2011). For a given fixed work flows matrix, the first problem is to solve to determine the location of machines and the distance between machines. Given the location of machines, the second problem can be addressed using fixed layout. The heuristic alternates between both problems until convergence is achieved. The obtained solution is not guaranteed to be optimal. In the respect to first problem since the flow volumes are assumed to be known, we encounter with the QAP. Accordingly, for real size problems, a heuristic solution procedure is required. Several heuristics have been considered for solving QAP, including pair-wise (Benjaafar and Sheikhzadeh, 2000), 2-opt heuristic (Lahmer and Benjaafar, 2005), multi-step exchange heuristics, genetic algorithms, simulated annealing (Castillo and Peters, 2003), and tabu search.

Interestingly as discussed in (Lahmer and Benjaafar, 2005), obtaining optimal solution in the design of distributed layouts is not crucial because of uncertainty in the value of design parameters in practise. Tangible examples are future demand and machine capacities. The inherent features of distributed layouts enable it to adapt to these uncertainties. Thus, near optimal solution are reasonably acceptable. (Solimanpur and Jafari, 2008) developed a branch-and-bound algorithm to optimally solve a nonlinear mixed integer model. the model integrates machine placement and flow assignment where un equal machines and continual representation of a distributed layout are considered. However, their algorithm is capable of solving small to medium size problems. As computational times exceed 24 hours for medium sized problem (15 machines, 5 products), authors suggest developing heuristic methods for solving large problem instances.

Simple improvement algorithm such as the 2-opt and 3-opt or CRAFT discussed in the literature have some limitations compared to some single point solution-based metaheuristic algorithms such as simulated annealing. A primary disadvantage of these local optimization algorithms is that they restrict their

search for optimal solution in downhill direction. In other words, the initial solution is changed only if the local search generates a neighborhood solution with lower value for minimization problems. To avoid entrapping in this local optima, the local search algorithm can be enhanced to allow it to back out of poor local optima regions and seek other regions. Thus, resulting in the higher quality solution. In addition, the local optimum solution for 2-opt (or 3-opt) depends on the local region in which the search occurs, which itself is function of the initial solution provided to it. The above argument gives the essence of the local search-based metaheuristic techniques such as simulated annealing and tabu search.

Hybrid Metaheuristic

It has become apparent that there is considerable advantage in using hybrid metaheuristic that are often significantly more efficient than pure metaheuristics in terms of running time and/or solution quality because of their built-in inherent synergy. A hybrid metaheuristic is a metaheuristic combined with one of problem-specific algorithms, simulations, exact techniques, heuristics, soft computing methods and other metaheuristics (Puchinger and Raidl, 2005). The majority of the publications in literature have turned to focused on problem-oriented approaches rather than algorithm-oriented. Therefore, developing an efficient hybrid approach is difficult task and requires specialized knowledge, expertise and skills in different areas of optimization (Blum *et al.*, 2006).

We begin with an overview of the classifications of strategies for combining metaheuristics and exact optimization techniques to provide background information needed to understand the solution procedures used in this study. Puchinger

and Raidl (2005) presented a survey of techniques that combine exact and metaheuristic and defined two categories: (1) collaborative combination, and (2) integrative combination. In the former, although the two algorithms exchange information, they are not part of each other and perform sequentially, intertwined, or in parallel. In the latter, one algorithm is embedded in another one. The resulting metaheuristic is either incorporated exact algorithm in metaheuristics or incorporated metaheuristic in exact method. An incorporated exact algorithm in metaheuristics either puts emphasis on local search approaches that are strengthened by the use of exact algorithms or exactly solves relaxed problems. Incorporated metaheuristic in exact method usually is used to determine bounds and incumbent solutions in branch and bound approaches. When dealing with local search using metaheuristic, the exact algorithm can assist search neighborhood and also allow to overcome local optima and to create high quality solution (Blum *et al.*, 2006). In other words, while a metaheuristic algorithm is able to find most promising regions in the solution space, exact methods can be incorporated within metaheuristic in order to find near-optimal solutions or optimal solutions in promising regions with reasonable time consumption. Exploiting problem specific knowledge can guide to how an exact method and metaheuristic are combined. For instance in mixed integer linear programming problem, where obtaining a feasible solution by local search methods may be very hard and the space of feasible solution might be very large, a metaheuristic assisted with an exact algorithm such as Mixed Integer Programming (MIP) or Linear Programming (LP) might be an applicable and promising algorithm. In addition, the availability of efficient general purpose MIP and LP-solvers such as IBM ILOG CPLEX, GUROBI, XPRESS, or the freely available SCIP and their relatively easy applicability makes this procedure particularly sensible in practice (Blum *et al.*, 2006).

Chapter 3

Mathematical Model for Distributed Layout Design

3.1. Introduction

In this chapter, we develop a mixed integer linear programming formulation for a dynamic distributed layout problem. Similar to the models found in the literature, the model is used to simultaneously determine machine placement and flow allocation. However, our model differs from preceding models in that: (1) we develop a model in mixed integer linear form to take advantage of the efficiency of the Simplex algorithm and minimize the time it takes to solve the problem to optimality, (2) production planning is incorporated in the model to simulate the real production environments, (3) lot streaming is considered to allow the process of splitting jobs, and (4) we approach workload balancing in a different way.

Managing the production resources and balancing them between successive time periods with the aim of minimizing the production costs is known as Production Planning (PP) (Safaei and Tavakkoli-Moghaddam, 2009). In a turbulent manufacturing environment, the established production planning and control also has to react to changes in demand, product mix or product process plans. In such

cases, a close link between the operational decision (i.e. machine sequencing and lot streaming) and execution activities (i.e. lot sizing and capacity planning) in PP and the change process is required. According to this understanding, integrating the concepts of dynamic distributed layout design and production planning is to be a crucial requirement to model and mimic the real production environments.

Lot streaming refers to the process of splitting a production into smaller jobs in order to move through several processing stages as quickly as possible (Baker, 1995). Therefore, different sublots of the same jobs can be processed simultaneously. By allowing the overlapping between successive operations, production may be significantly accelerated and a reduction on the work-in-process inventory levels may be also obtained. A sublot may contain several items. Therefore, after some items in each sublot are completed in a machine, they have to wait until all of the items in the sublot are finished before they move to the next machine. Those products waiting for others to be completed are part of the work-in-process inventory. Lot streaming also improves customer service, since partially completed sublots may be delivered before the whole job (order) is completed (Potts and Van Wassenhove, 1992).

The majority of approaches that address distributed layout design problems tend to minimize material handling costs only. These include Benjaafar and Sheikhzadeh (2000); Baykasoglu (2003) and Lahmer and Benjaafar (2005). However, when systems reconfiguration and production planning are considered concurrently, the actual problem involves other costs associated with machine relocation, setup, inventory holding, in-house production and subcontracting needs (See Tables 3.1 and 3.2 for a comparison between this paper and recently published articles on distributed layout).

In this thesis, the concept of resource elements (REs) is innovatively used as basis to impose workload balancing between machines of varying degrees of capabilities. In most previous studies considering workload balancing, a workload

Table 3.1: List of manufacturing attributes

1. Alternative Routing
2. Demand Fluctuation
3. Dynamic System Reconfiguration
4. Workload Balancing
5. Lot-Splitting
6. Types of Tools Required by a Part
7. Types of Tools Available on a Machine
8. Production Planning
9. Setup Cost
10. Movement of Parts (Material Handling Cost)
11. Machine Capacity
12. Subcontracting Cost
13. Operation Cost

Table 3.2: Attributes used in the present study and in a sample of recently published articles

Article/Attributes	1	2	3	4	5	6	7	8	9	10	11	12	13
Present study	×	×	×	×	×	×	×	×	×	×	×	×	×
Nageshwaraniyer *et al.* (2013)		×	×							×			
Hamedi *et al.* (2012)		×				×	×			×	×		
Lahmer and Benjaafar (2005)		×	×							×	×		×
Baykasoglu (2003)		×				×	×			×	×		
Urban *et al.* (2000)		×								×	×		
Benjaafar and Sheikhzadeh (2000)		×	×							×	×		×

Note: Attributes' names are referred in Table 3.1

is required to be evenly divided among machines that are deemed similar (may not be identical). In our work, we approach workload balancing in a different way. In our model, workload balancing is: (1) fully enforced among identical machines, (2) partially enforced among machines having some shared capabilities, and (3) not totally enforced among dissimilar machines. The aim of this Chapter is to give more details on our mathematical model and discuss some features of it.

3.2. Problem Description

Consider a manufacturing system processing P products in T number of equal planning periods where the demand for the products may vary from period to period deterministically. The system consists of M machines to be distributed over N distinct locations ($N = M$) and reconfiguration may take place at the beginning of each planning period. There are a total of R resource elements, and each machine has some of these REs, representing the capabilities it shares with other machines, as well as those that are unique to it. Processing a part requires a set of operations to be performed in a given sequence. A particular operation can be performed using a given resource element, and machines possessing this element are considered as alternative routes for this operation. The processing time for each operation is known. In a given time period, a demand for a part can be satisfied by producing it in-house, subcontracting its production, or using inventory carried over from the previous period. Without loss of generality, we assume a part inventory is zero at the beginning of the first period and at the end of the last period. A production lot of a part may be split into smaller sublots that are to be processed independently. The material flow cost of a part is linearly related to the distance it travels using the material handling system. The cost to relocate a machine is also assumed to be linearly related to the relocation distance. However, we assume that the distance between a pair of locations when moving a part is not the same as the distance between the same pair of locations when relocating a machine. This is because parts are moved using a material handling system (e.g., AGV with a specified path), whereas machines are relocated in a different way. The workload of the system in a given time period is evenly distributed among the machine tools that share the particular resource element being used. The overall objective is to minimize the total costs associated with material handling, machine relocation, subcontracting, setup, inventory holding and internal part production.

3.3. Notation

The problem described in the previous Section is formulated as a mixed integer linear programming. The notations used in this formulation are presented below.

Indexes and Input Data:

T Number of equal planning periods where planning periods are indexed by $t = 1, 2, ..., T$.

P Number of products where products are indexed by $p = 1, 2, ..., P$.

O_p Number of operations required by product p where operations are indexed by $o = 1, 2, ..., O_p$.

N_p Maximum number of sublots of product p in a given time. period where production sublots are indexed by $n = 1, 2, ..., N_p$.

M Number of machines in the manufacturing facility where machines are indexed by $m = 1, 2, ..., M$.

R Number of resource elements in the manufacturing facility where resource elements are indexed by $r = 1, 2, ..., R$.

L Number of locations at which machines are installed, where locations are indexed by $l = 1, 2, ..., L$.

J Number of groups of machines with similar functionality where groups are indexed by $j = 1, 2, ..., J$.

C Length of a planning period in terms of available work time in minutes.

$D_{p,t}$ Demand quantity for product p in time period t.

Θ_p Unit cost of producing product p in-house (not including setup).

$\hat{\Theta}_p$ Unit cost of subcontracting product p.

H_p Unit inventory holding cost per period for product p.

F_p Material handling cost per unit distance for one unit of product p.

$U_{o,p}$ Unit processing time for operation o of product p.

$A_{r,m}$ A binary datum which equals 1 if resource element r is available on machine m; 0 otherwise.

$B_{r,o,p}$ A binary datum which equal to 1 if resource element r is required by operation o of product p; 0 otherwise. An operation requires only a single resource element and machines having this resource element are considered as alternative routing for this operation.

$K_{o,p,m}$ A binary datum which equals 1 if operation o of product p can be processed on machine m; 0 otherwise. $K_{o,p,m} = \sum_{r=1}^{R}(A_{r,m} \times B_{r,o,p})$.

$E_{l,l'}$ Machine relocation distance between locations l and l'.

$\tilde{E}_{l,l'}$ Material handling distance between locations l and l'.

G_m Relocation cost per unit distance for machine m.

S_p Setup cost for processing a sublot of product p.

Ω Large positive number.

Variables:

Continuous Variables:

$v_{p,t}$ Production lot size of product p in time period t.

$b_{n,p,t}$ The size of the n^{th} sublot of product p in time period t.

$\hat{v}_{p,t}$ Volume of product p subcontracted in time period t.

$\delta_{o,n,p,m,t}$ The time elapsed in processing operation o of the n^{th} sublot of product p on machine m in time period t.

$h_{p,t}$ Inventory level of product p at the beginning of period t.

$d_{o,n,p,t}$ Distance between the locations where operations o and $o+1$ of n^{th} sublot of product p are processed multiplied by the sublot size $b_{n,p,t}$ in time period t.

$e_{m,t}$ Distance between the location of machine m in period $t-1$ and its location in period t.

Binary Variables:

$\alpha_{m,l,t}$ A binary variable equal to 1 if machine m is located at location l in time period t; 0 otherwise.

$\gamma_{o,n,p,m,t}$ A binary variable equal to 1 if operation o of the n^{th} sublot of product p is processed by machine m in time period t; 0 otherwise.

$y_{n,p,t}$ A binary variable equal to 1 if n^{th} sublot of product p is created and processed in time period t; 0 otherwise.

3.4. Objective Function and Constraints

Following the problem description and notation given in Sections 3.2 and 3.3, the comprehensive mathematical model for distributed layout manufacturing system design is presented below.

Minimize:

$$\begin{aligned} Z &= \sum_{t=2}^{T}\sum_{m=1}^{M}(G_m \cdot e_{m,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{o=1}^{O_p-1}(F_p \cdot d_{o,n,p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(H_p \cdot h_{p,t}) \\ &+ \sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{n=1}^{N_p}(S_p \cdot y_{n,p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(\Theta_p \cdot v_{p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(\hat{\Theta}_p \cdot \hat{v}_{p,t}) \end{aligned} \tag{3.1}$$

Subject to:

$$e_{m,t} \geq E_{l,l'} + \Omega(\alpha_{m,l,t-1} + \alpha_{m,l',t}) - 2\Omega;$$

$$\forall(m,t,l,l')|t > 1 \tag{3.2}$$

$$e_{m,t} \leq E_{l,l'} - \Omega(\alpha_{m,l,t-1} + \alpha_{m,l',t}) + 2\Omega;$$

$$\forall(m,t,l,l')|t > 1 \tag{3.3}$$

$$d_{o,n,p,t} \geq \widetilde{E}_{l,l'} \cdot b_{n,p,t} + \Omega\left(\alpha_{m,l,t} + \gamma_{o,n,p,m,t} + \alpha_{m',l',t} + \gamma_{o+1,n,p,m',t}\right) - 4\Omega;$$

$$\forall(o,n,p,t,m,m',l,l')|(o < O_p \ \&\ K_{o,p,m} \times K_{o+1,p,m'} = 1) \tag{3.4}$$

$$d_{o,n,p,t} \leq \widetilde{E}_{l,l'} \cdot b_{n,p,t} - \Omega\left(\alpha_{m,l,t} + \gamma_{o,n,p,m,t} + \alpha_{m',l',t} + \gamma_{o+1,n,p,m',t}\right) + 4\Omega;$$

$$\forall(o,n,p,t,m,m',l,l')|(o < O_p \ \&\ K_{o,p,m} \times K_{o+1,p,m'} = 1) \tag{3.5}$$

$$v_{p,1} + \hat{v}_{p,1} = D_{p,1} + h_{p,2}; \forall(p) \tag{3.6}$$

$$v_{p,t} + h_{p,t} + \hat{v}_{p,t} = D_{p,t} + h_{p,t+1}; \forall(p,t)|(1 < t < T) \tag{3.7}$$

$$v_{p,T} + h_{p,T} + \hat{v}_{p,T} = D_{p,T}; \forall p \tag{3.8}$$

$$\sum_{p=1}^{P} \sum_{n=1}^{N_p} \sum_{o=1}^{O_p} \delta_{o,n,p,m,t} \leq C; \forall (m,t) \tag{3.9}$$

$$\delta_{o,n,p,m,t} \geq U_{o,p} \cdot b_{n,p,t} + \Omega \cdot (\gamma_{o,n,p,m,t} - 1); \forall (o,n,p,m,t) | (K_{o,p,m} = 1) \tag{3.10}$$

$$\delta_{o,n,p,m,t} \leq U_{o,p} \cdot b_{n,p,t} - \Omega \cdot (\gamma_{o,n,p,m,t} - 1); \forall (o,n,p,m,t) | (K_{o,p,m} = 1) \tag{3.11}$$

$$\delta_{o,n,p,m,t} \leq \Omega \cdot \gamma_{o,n,p,m,t}; \forall (o,n,p,m,t) | (K_{o,p,m} = 1) \tag{3.12}$$

$$\gamma_{o,n,p,m,t} \leq K_{o,p,m}; \forall (o,n,p,m,t) \tag{3.13}$$

$$\sum_{p=1}^{P} \sum_{n=1}^{N_p} \sum_{o=1}^{O_p} B_{r,o,p} \times \delta_{o,n,p,m,t} \geq \left(\frac{\sum_{m''=1}^{M} \sum_{p=1}^{P} \sum_{n=1}^{N_p} \sum_{o=1}^{O_p} B_{r,o,p} \times \delta_{o,n,p,m'',t}}{\sum_{m'=1}^{M} A_{r,m'}} \right) \times \Upsilon; \forall (r,m,t) \tag{3.14}$$

$$\sum_{m=1}^{M} \gamma_{o,n,p,m,t} = y_{n,p,t}; \forall (o,n,p,t) \tag{3.15}$$

$$b_{n,p,t} \leq \Omega \cdot y_{n,p,t}; \forall (n,p,t) \tag{3.16}$$

$$\sum_{n=1}^{N_p} b_{n,p,t} = v_{p,t}; \forall (p,t) \tag{3.17}$$

$$\sum_{l=1}^{L} \alpha_{m,l,t} = 1; \forall (m,t) \tag{3.18}$$

$$\sum_{m=1}^{M} \alpha_{m,l,t} = 1; \forall (l,t) \tag{3.19}$$

$$\alpha_{m,l,t},\ \gamma_{o,n,p,m,t},\ y_{n,p,t} \text{ are binary.} \tag{3.20}$$

The objective function in Eq. (3.1) consists of six cost terms: machine relocation, material handling, inventory holding, machine setup, in-house production, and subcontracting needs in that order. The constraints in Eqs. (3.2) and (3.3) are to equate the variable $e_{m,t}$ to the distance $E_{l,l'}$ if machine m is relocated from location l to location l' at the beginning period t. The value of the variable $d_{o,n,p,t}$ is equal to the product $\widetilde{E}_{l,l'} \cdot b_{n,p,t}$ if operations o and $o+1$ of n^{th} sublot of product p are processed on machines m at location l and m' at location l', respectively, in period t. This requirement is enforced by Eqs. (3.4) and (3.5). The constraints in Eqs. (3.6), (3.7) and (3.8) are for inventory balance. Eq. (3.9) guarantees that the workload on machine m in time period t is less or equal to the available time C. Eqs. (3.10) and (3.11) state that the time $\delta_{o,n,p,m,t}$ elapsed in processing operation o of the n^{th} sublot of product p on machine m in time period t is equal to the product $U_{o,p} \cdot b_{n,p,t}$ if this operation is assigned to this machine in this time period. Otherwise, the value of this variable is set to zero by Eq. (3.12). The constraint in Eq. (3.13) permits the processing of operation o of sublot n of product p on machine m in time period t if and only if operation o of product p can be assigned on machine m. The workload balancing constraint is in Eq. (3.14). The left-hand side of this equation is the amount of workload performed by machine m in period t using resource element r. The right-hand side of this constraint is expressed as (i) the total workload of all the machines using resource element r which equal to $\sum_{m''=1}^{M} \sum_{p=1}^{P} \sum_{n=1}^{N_p} \sum_{o=1}^{O_p} B_{r,o,p} \times \delta_{o,n,p,m'',t}$, (ii) divided by the number of machines having this resource element $\sum_{m'=1}^{M} A_{r,m'}$, and (iii) multiplied by a factor $\Upsilon \in (0,1)$. If this factor is set very close to 1, the workload of the system in using resource element r will be evenly distributed among the

machines having this resource element. Eq. (3.15) ensures the assignment of the o^{th} operation of the n^{th} sublot of part p in time period t to one of the machines if the sublot is created. The constraint in Eq. (3.16) ensures that the production quantity of each sublot in each time period, $b_{n,p,t}$, is equal to 0 if this sublot is not created (i.e. $y_{n,p,t} = 0$) . The constraint in Eq. (3.17) enforces that the sum of the sizes of the sublots of a given product should be equal to the production lot size of that particular product in each period. The constraints in Eqs. (3.18) and (3.19) ensure that each location is assigned to only one machine and each machine is assigned to only one location. Eq. (3.20) is the integrality constraint on the binary variables.

Chapter 4

The proposed algorithms

4.1. Introduction

Most facility design and planning problems are NP-hard (see list given by Garey & Johnson (1979), pp. 236-244). These problems cannot be solved optimally within polynomial computational time and they usually require NP (exponential) time. Thus, it is of great interest to be able to give near-optimal solutions in a reasonable amount of time. Since the proposed mathematical model is an enlarged version of facility layout design with several added features, we can conclude that it is also NP- complete. In order to solve this model, particularly for larger problem sizes, we developed three solution procedures: (1) a Linear Programming Embedded Simulated Annealing (LPSA), (2) a Pure Simulated Annealing (PSA), and (3) a Linear Programming Embedded Genetic algorithm (LPGA). These simulated annealing employs multiple search paths which may result in effective search of the solution space. The concept of embedding linear programming into the simulated annealing is similar to the work presented in Teghem *et al.* (1995) and (Defersha and Chen, 2008).

4.2. Linear Programming Embedded Simulated Annealing

Simulated Annealing (SA) is a general search algorithm which was first introduced by Kirkpatrick *et al.* (1983) for solving hard combinatorial optimization problems. It is so named because of its analogy to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration, and is thus free of crystal defects. If the cooling schedule is sufficiently slow, the final configuration results in a solid with such superior structural integrity. Simulated annealing establishes the connection between this type of thermodynamic behaviour and the search for a global minima for an optimization problem. Furthermore, it provides an algorithmic means for exploiting such a connection. To describe this algorithmic feature of SA, lets start by defining X_n as the solution of the optimization problem at the n^{th} iteration. At this iteration, SA generates a neighborhood solution X'_n by applying systematically designed move operators. Two values of the objective function $f(X_n)$ and $f(X'_n)$ are evaluated corresponding to the current solution and the newly selected solution, respectively. The candidate solution X'_n is then accepted as the current solution based on the acceptance probability P given in Eq. (4.1). The parameter H_n in this equation is the temperature at the n^{th} iteration which approaches to zero as n increases. As it can be seen from Eq. (4.1), the probability of acceptance of non-improving solution is higher at the early stage of the iteration when the temperature is high. As the algorithm proceeds and the temperature approaches to zero, hill-climbing moves occur less frequently and the solution converges.

$$P(\text{Accept } X'_n \text{ as next solution}) = \begin{cases} 1 & ; \text{ if } f(X'_n) \leq f(X_n) \\ \exp\left[\frac{f(X_n)-f(X'_n)}{H_n}\right] & ; \text{ if } f(X'_n) > f(X_n) \end{cases} \tag{4.1}$$

In doing the procedure described in the previous paragraph, SA visits a sequence of random solutions $\{X_1, X_2, \cdots, X_n, X_{n+1}, \cdots, X_N\}$ where a single solution is visited at each iteration. We call this single search path simulated annealing (SSP-SA). However, as pointed out in Lee and Lee (1996), following a single search path may not be necessary from performance point of view. A multiple search path SA (MSP-SA) performs S independent versions of simulated annealing using the same search space, neighborhood generation and cooling schedule. Each one of these independent versions stops after N iterations to provide S independent terminal solutions $\{X_{N,1}, X_{N,2}, \cdots, X_{N,S}\}$. Then out of these terminal solutions, the best one is chosen as the final solution X_N. For better performance, the search paths may be allowed to interact periodically. Such interaction can update each search path after a given number of iterations by the best solution (X_{Best}) so far found in the entire search process. A general pseudocode of MSP-SA with such interaction among the search paths is given in Figure 4.1. This Pseudocode has been used as a template for the implementation of the algorithm developed in this thesis. In this Pseudocode, the interaction of the search paths happens every I number of iterations (see line 27). Cooling is performed every Q iterations using the equation $H_{r+1} = \alpha \times H_r$ (see line 31) where $\alpha \in (0, 1)$ is the cooling coefficient generally chosen to be close to 1 and r is the index of the temperature level. The complete set of notations used in the pseudocode is given below and these notations are totally sperate from those used in the mathematical model and should not be confused.

Notations used in the Pseudocode

j	Index of search path; $j = 1, 2, \cdots, S$ where S is the number of paths followed.
n	Overall iteration counter; $n = 1, 2, ..., N$ where N is the maximum number of iterations in each search path.
$X_{n,j}$	The solution at the n^{th} iteration along the j^{th} search path.
$X'_{n,j}$	The neighborhood solution at the n^{th} iteration along the j^{th} search path.
ρ	The random number generated for making a stochastic decision for the new solution.
r	Index for the temperature levels in the cooling schedule.
α	Cooling schedule exponent.
H_r	Temperature at the r^{th} level, $H_r = \alpha \times H_{r-1} = \alpha^{r-1} \times H_1$
q	Iteration counter at each temperature level; $q = 1, 2, \cdots, Q$ where Q is the number of iterations to be performed in each search path at each temperature level
X_{Best}	Best solution found so far in the entire search process.
I	Number of iterations to be performed between interactions of search paths.

4.2.1. Search Space and Solution Encoding

A search space of a heuristic algorithm is a set of solution points that can be potentially visited by the search process. Determining appropriate search space is

```
MSP-SA()    //Multiple Search Path SA
{
1     Randomly generate S solution points {X_{1,1}, X_{1,2}, ..., X_{1,S}}
2     Set initial temperature T_1
3     Set n = 1, r = 1;
4     Set X_Best = X_{1,1}
5     REPEAT
6     {
7           FOR q = 1 to Q       //Q = number of iterations at temperature T_r
8           {
9                 FOR j = 1 to S       //S = number of independent search paths
10                {
11                      Generate X'_{n,j} from X_{n,j}
12
13                      IF f(X'_{n,j}) ≤ f(X_{n,j})
14                            X_{n+1,j} = X'_{n,j}
15                      ELSE IF exp((f(X_{n,j}) − f(X'_{n,j}))/H_r) > ρ
16                            X_{n+1,j} = X'_{n,j}
17                      ELSE
18                            X_{n+1,j} = X_{n,j}
19
20
21                      IF f(X_{n+1,j}) < f(X_Best)   //Update X_Best
22                            X_Best = X_{n+1,j}
23                }
24                n = n + 1
25                IF (n mod I) = 0       //Update search paths
26                {
27                      Set X_{n,j} = X_Best for j = 1, ··· S
28                }
29          }
30          r = r + 1
31          H_r = α × T_{r−1}       //Cooling every Q iterations.
32    }
33    UNTIL n = N
}
```

Figure 4.1: Pseudocode for an instance of a MSP-SA with interacting paths

the first most important step in applying a metaheuristic algorithm. It influences the implementation and the performance of the algorithm. The search space explored in solving the proposed model is a set of solution points in which a solution point can be uniquely addressed using a valid combination of values of the integer variables $\{y_{n,p,t},\ \alpha_{m,l,t}$ and $\gamma_{o,n,p,m,t}$ for all $(o,n,p,m,t)\}$. In this regard, we define a valid combination of the values of these integer variables as a combination that satisfy the constraints composed of only the integer variables. These constraints are in Eqs. (3.13), (3.15), (3.18), (3.19), and (3.20). The values of the continuous variables which optimally correspond to a given combination of the integer variables are determined by solving a linear programming (LP) subproblem. The solution of the LP-subproblem satisfies the remaining sets of constraints by having the continuous variables in their equations.

To explore the search space described above, a solution encoding is developed that can be decoded to give a valid combination of the values of the integer variables. This solution encoding is given in Figure 4.2. In this Figure, the solution encoding is detailed in four levels. In Level-I, it is shown that the solution encoding is composed of T main segments, one for each planning period. Level-II details the segment corresponding to $t = 1$. This segment is divided into right-hand and left-hand segments (LHS- and RHS-segments). The LHS-segment determines the allocation of M machines to $L = M$ locations in a given period t. In this segment, $M_{l,t}$ takes the index of the machine placed at location l in time period t which can be used to decode the value of the integer variable $\alpha_{m,l,t}$ using Eq. (4.2). Since $M_{l,t}$ takes only a single value $m \in (1, ..., M)$, the decoded values of $\alpha_{m,l,t}$ $\forall(m,l,t)$ satisfies the constraint in Eq. (3.19). Moreover, the values of $M_{l,t}$ in the LHS-segment (for a given t) is unique, i.e. $M_{l,t} \neq M_{l',t}$ for any pair of locations l and l' (such that $l \neq l'$). This guarantees the satisfaction of the constraint in Eq. (3.18).

The RHS-segment encodes the decision variables $y_{n,p,t}$ and $\gamma_{o,n,p,m,t}$. This

segment has P subsegments corresponding to each product. The subsegment corresponding to $p = 1$ has been detailed in Level-III. This subsegment has in turn N_p subsegments corresponding to each sublot of product p. The subsegment corresponding to $n = 1$ is detailed at Level-IV. In this level $y_{n,p,t}$ is directly encoded and takes a value in $\{0, 1\}$ to indicate whether or not sublot n of product p is created in time period t. The element $M_{o,n,p,t}$ takes the index of one of the machines that can process the o^{th} operation of part p. In other words, $M_{o,n,p,t}$ takes one of the values in $\{m|K_{o,p,m} = 1\}$. Having the values for $y_{n,p,t}$ and $M_{o,n,p,t}$, the values of the variable $\gamma_{o,n,p,m,t}$ can be decoded using the Eq. (4.3). This decoding procedure ensures that the constraints in Eqs. (3.13) and (3.15) are satisfied.

$$\alpha_{m,l,t} = \begin{cases} 1 \;\; ; \;\; \text{If } M_{l,t} = m \\ \\ 0 \;\; ; \;\; \text{Otherwise} \end{cases} \tag{4.2}$$

$$\gamma_{o,n,p,m,t} = \begin{cases} y_{n,p,t} \;\; ; \;\; \text{If } M_{o,n,p,t} = m \\ \\ 0 \qquad ; \;\; \text{Otherwise} \end{cases} \tag{4.3}$$

4.2.2. Linear Programming Subproblem

In the previous Section, it was stated that the values of the continuous variables which optimally correspond to a given combination of the integer variables are determined by solving a LP-subproblem. In this Section we present how this LP-subproblem is formulated. The constraints in Eqs. (3.2) and (3.3) in the main model were used to set $e_{m,t} = E_{l,l'}$ if the variables $\alpha_{m,l,t-1} = \alpha_{m,l',t} = 1$. However, in formulating the LP-subproblem corresponding to a given solution point, the integer variables are known. Hence, the value of $e_{m,t}$ can be computed

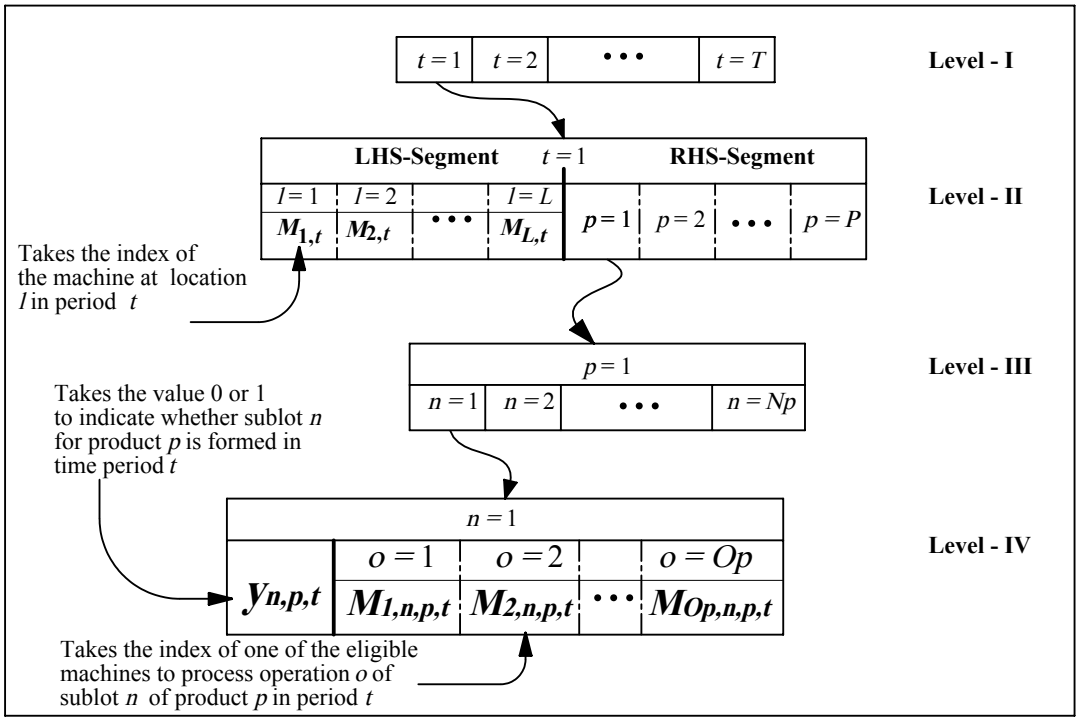


Figure 4.2: Solution representation used in LPSA and LPGA

before solving the LP-subproblem. This makes the constraints in Eqs. (3.2) and (3.3) unnecessary in the LP-subproblem. Moreover, knowing the values of $e_{m,t}$ for all $(m,\ t)$, the machine relocation cost term can be removed from the objective function of the LP-subproblem as this term can be computed before solving the LP-subproblem. The objective function of the LP-subproblem is given in Eq. (4.4). The overall objective function $f(X)$ of a trial solution X in the simulated annealing is then equal to the machine relocation cost plus the objective function of the LP-subproblem. The constraints in Eqs. (3.4) and (3.5) of the main model are used to set $d_{o,n,p,t} = E_{l,l'} \cdot b_{n,p,t}$ if the sum of the binary decision variables $\alpha_{m,l,t}$, $\gamma_{o,n,p,m,t}$, $\alpha_{m',l',t}$, and $\gamma_{o+1,n,p,m',t}$ is equal to four. Knowing the values of these binary variables, in formulating the LP-subproblem, these two constraints can be replaced by a single equality constraint $d_{o,n,p,t} = E_{l,l'} \cdot b_{n,p,t}$ for every $(\alpha_{m,l,t} + \gamma_{o,n,p,m,t} + \alpha_{m',l',t} + \gamma_{o+1,n,p,m',t} = 4)$ as shown in Eq. (4.5). Following a

similar logic, the constraints in Eqs. (3.10) and (3.11) can also be replaced by a single equality constraint in the LP-subproblem as shown in Eq. (4.6). The constraint in Eq. (3.12) was required to set $\delta_{o,n,p,m,t} = 0$ if the binary variable $\gamma_{o,n,p,m,t} = 0$. In the LP-subproblem, this constraint is not required since the continuous variable $\delta_{o,n,p,m,t}$ corresponding to $\gamma_{o,n,p,m,t} = 0$ can be excluded from the LP formulation. The constraints in Eqs. (3.6)-(3.9), (3.14), (3.16), and, (3.17) of the main model are also applicable in the LP-subproblem. Whereas, the constraints in Eqs. (3.13), (3.15), and (3.18)-(3.20) are not required in the LP-subproblem as these constraints are composed of only the integer variables and are taken care of by the solution representation. The complete LP-subproblem is given below.

LP: given $(\alpha_{m,l,t},\ \gamma_{o,n,p,m,t},\ y_{n,p,t})$ for all (o,n,p,m,t)

Minimize:

$$\begin{aligned} Z_{LP} \;=\; & \sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{o=1}^{O_p-1}(F_p \cdot d_{o,n,p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(H_p \cdot h_{p,t}) \\ + \; & \sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{n=1}^{N_p}(S_p \cdot y_{n,p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(\Theta_p \cdot v_{p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(\hat{\Theta}_p \cdot \hat{v}_{p,t}) \end{aligned} \tag{4.4}$$

Subject to:

$$d_{o,n,p,t} = E_{l,l'} \cdot b_{n,p,t};$$

$$\forall(o,n,p,t,m,l,m',l')\,|\,(\alpha_{m,l,t} + \gamma_{o,n,p,m,t} + \alpha_{m',l',t} + \gamma_{o+1,n,p,m',t} = 4) \tag{4.5}$$

$$\delta_{o,n,p,m,t} = U_{o,p} \cdot b_{n,p,t};$$

$$\forall(o,n,p,m,t)|\gamma_{o,n,p,m,t}=1; \tag{4.6}$$

and

Eqs.(3.6), (3.7), (3.8), (3.9), (3.14), (3.16), **and**, (3.17)

4.2.3. Implementation Challenge

A single iteration cycle in a particular search direction of the proposed multiple search path SA is shown in Figure 4.3. From this Figure it can be seen that the LP-subproblem has to be formulated (in step 3) and solved (in step 4) for each trial solution visited by the algorithm. In our first attempt to implement the LP-embedded solution approach, we encountered a computational difficulty as the algorithm needs substantial amount of time to complete the evaluation of a trial solution. At first it was believed that the simplex algorithm in step 4 was the source of the computational hurdle. However, a closer look into the evaluation subroutine reveals that the LP-Subproblem formulation process (step 3) was the main source of the computational difficulty. In particular, the process of adding the constraint in Eq. (4.5) into the ILOG-CPLEX modelling environment was very time consuming. In its current form, this constraint can be added into the modeling environment following the pseudocode in Figure 4.4. From this Pseudocode, it can be seen that an enormous amount of looping is required to insert the equality constraint into the model. For instance, assuming each part has O number of operations and N number of sublots, the number of FOR-loops required to insert this constraint is $P \cdot O \cdot N \cdot T \cdot L^2 \cdot M^2$.

To alleviate the computational difficulty stated above, we reformulate the LP-subproblem in a systematic way. First, we define $D_{n,p,t}$ as the distance traveled by a unit of item in sublot n of product p in time period t. This value can be computed using the information that can be obtained from a trial solution being

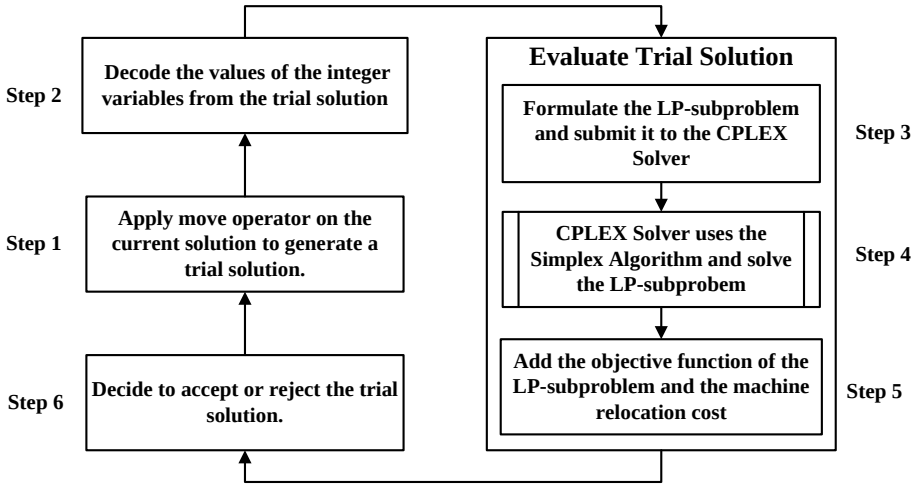


Figure 4.3: One complete iteration in a given search path of the SA

evaluated and by applying the pseudocode in Figure 4.5. In lines 4 and 5, the location of machine m in time period t is obtained from the trial solution X' and stored in a matrix Location$[m][t]$. In line 19, the index of the machine assigned to processes operation (o, n, p) in period t is obtained from X' and stored in m, and the location of this machine is stored in l (in line 20). The same is done in lines 22 and 23 for the operation $(o + 1, n, p)$ to get the machine index m' and the location l' where this operation is to be processed in period t. The distance traveled $\tilde{E}_{l.l'}$ by the part in getting these two consecutive operations done is recursively added to $D_{n,p,t}$ in line 25. Once this value is calculated for every (n, p, t), the material handling cost term (first term in Eq. (4.4)) of the LP-subproblem can be replaced by an equivalent term $\sum_{t=1}^{T} \sum_{p=1}^{P} \sum_{n=1}^{N_p} (F_p \cdot D_{n,p,t} \times b_{n,p,t})$. This makes the constraints in Eq. (4.5) and its implementation shown in Figure 4.4 unnecessary. Assuming each part has O number of operations and N number of sublots, the number of FOR-loops in the pseudocode in Figure 4.5 is only $[(T \cdot L) + (P \cdot O \cdot N \cdot T)]$. This is much lower than the number of loops in Figure 4.4. For example, if we assume $P = 40$, $O = 8$, $N = 3$, $T = 4$, and $M = L = 20$, the number of loops in Figure 4.4 is 6.144×10^8 whereas that in Figure 4.5 is only 3,920. With this adjustment, it was possible to reduce the execution time

```
1   FOR[1]  o = 1 to o = O_p − 1
2     FOR[2]  n = 1 to N_p
3       FOR[3]  p = 1 to P
4         FOR[4]  t = 1 to T
5           FOR[5]  m = 1 to M
6             FOR[6]  m' = 1 to M
7               FOR[7]  l = 1 to L'
8                 FOR[8]  l' = 1 to L
9
10                  IF (α_{m,l,t} + γ_{o,n,p,m,t} + α_{m',l',t} + γ_{o+1,n,p,m',t} = 4)
12                  ADD CONSTRAINT  d_{o,n,p,t} = Ẽ_{l,l'} · b_{n,p,t}
13
13                END FOR[8]
14              END FOR[7]
15            END FOR[6]
16          END FOR[5]
17        END FOR[4]
18      END FOR[3]
19    END FOR[2]
20  END FOR[1]
```

Figure 4.4: Pseudocode for the formulation of the constraint in Eq. (4.5) of the LP-subproblem

of a single iteration cycle shown in Figure 4.3 from several minutes to just a fraction of a second. This adjustment makes the proposed LP-embedded simulated annealing a working algorithm.

4.2.4. Two Search Phases

Obtaining a good static layout may be simpler than obtaining a dynamic layout which optimizes system reconfiguration along other cost terms. This is because the search space for dynamic layout is much wider than that of static layout. On the other hand, since excessive reconfiguration is not likely, the best layout with dynamic reconfiguration could be a neighborhood solution to the best configuration with static configuration. With this consideration, in the proposed algorithm, the search process is divided into two phases. In the first phase, the algorithm attempts to find the best static configuration. Whereas, in the second phase the algorithm attempts to find an optimal reconfiguration based on the

```
1   FOR[1]   t = 1 to t = T
2       FOR[2]   l = 1 to l = L
3
4           m = X'[t][LHS-segment][l][M_{l,t}][a]
5           Location[m][t] = l
6
7       END FOR[2]
8   END FOR[1]
9   ..........................................................................................
10
11  FOR[3]   t = 1 to t = T
12      FOR[4]   p = 1 to p = P
13          FOR[5]   n = 1 to n = N_p
14
15              D_{n,p,t} = 0 //Initialize to zero
16
17              FOR[6]   o = 1 to o = O_p − 1
18
19                  m = X'[t][RHS-segment][p][n][M_{o,n,p,t}][b]
20                  l = Location[m][t]
21
22                  m' = X'[t][RHS-segment][p][n][M_{o+1,n,p,t}]
23                  l' = Location[m'][t]
24
25                  D_{n,p,t} = D_{n,p,t} + Ẽ_{l,l'}
26
28              END FOR[6]
29          END FOR[5]
30      END FOR[4]
31  END FOR[3]
```

[a] $X'[t]$[LHS-segment]$[l][M_{l,t}]$ means trial solution X', segment t at level-I, LHS-segment at Level-II, location l, machine index $M_{l,t}$ (see Figure 4.2)

[b] $X'[t]$[RHS-segment]$[p][n][M_{o,n,p,t}]$ means trial solution X', segment t at level-I, RHS-segment at Level-II, product p, sublot n, operation o, machine index $M_{o,n,p,t}$ (see Figure 4.2)

Figure 4.5: Pseudocode to calculate the distance traveled by a part

solution found in the first phase. Dividing the search process into these phases is accomplished as follows. First, initial solutions without system reconfiguration generated along each search direction. In this regard, a solution with static configuration has all of its LHS-segments identical to represent identical layout configurations across all periods. Then only move operators that do not result in system reconfiguration are applied during the first phase of the search. Finally, starting from the solution of the first phase, the search continues by applying move operators that can result in system reconfiguration. These move operators are described in the next Section.

4.2.5. Perturbation Operators

In the proposed SA algorithm, we use four solution perturbation operators that are applied at each iteration with small probabilities. These operators can be classified into two categories: (1) layout perturbation operators and (2) product flow perturbation operators. The layout perturbation operators are applied on the LHS-segments of a solution to affect its machine configuration/reconfiguration. These operators are Machines Locations Swap Operator (MLSO) and Machine Locate and Fix Operator (MLFO). The MLSO operator is applied with small probability β_1 in two different manners depending on the phase of the search. In the first phase of the search, the operator first arbitrarily selects two locations l and l'. Then, for these pair of locations, it swaps the values of $M_{l,t}$ and $M_{l',t}$ of the LHS-segment of each period of the solution. Applied in this manner, the operator keeps the LHS-Segments identical across the periods and results in a trial solution with static configuration. Whereas in doing such swaps in the second phase, MLSO selects the pair of locations l and l' for each period independently resulting in non-identical LHS-Segments of the affected solution. The implementation of this perturbation operator is shown using pseudocode in Figure 4.6. The second layout perturbation operator, MLFO, is applied only in

the second phases of the search. It fixes an arbitrarily chosen machine (m^*) at an arbitrarily chosen location (l^*) between arbitrarily chosen time periods t_1 and t_2 (such that $t_1 < t_2 \leq T$) with a probability β_2. The essence of this operator is that once a particular machine is moved to a particular location, it is more likely for this machine to remain in that particular location for a certain number of time periods. This operator is implemented following the pseudocode in Figure 4.7. The product flow perturbation operators are applied on RHS-Segments of a solution in both the first and second phase of the search to alter the flow of the products. These operators are Sublot Flip Operator (SFO) and Alternative Machine Swap Operator (AMSO). The SFO is applied to each of the $y_{n,p,t}$ of the RHS-Segments with a small probability β_3 to switch their value between 0 and 1. The operator AMSM is applied on each $M_{o,n,p,t}$ in the RHS-segments of a solution to alter its value to one of machines that can process operation o of product p. It is applied on each $M_{o,n,p,t}$ with small probability β_4.

4.2.6. Steps of the LPSA

As previously mentioned, the LPSA uses multiple search paths, which correspond to the individuals of a simulated annealing algorithm. The steps of LPSA and notation used are presented in below and in Figure 4.8.

s	Index of search paths, $s = 1,\ 2,\ ...,\ S$ where S is the maximum number of search paths
n	Iteration counter, $n = 1,\ 2,\ ...,\ N$ where N is the maximum number of iterations in each search path
X_{ns}	The solution at the n^{th} iteration along the s^{th} search path
α	Cooling schedule exponent
r	Index for the temperature levels in the cooling schedule
T_r	Temperature at the r^{th} level, $T_r = \alpha \times T_{r-1} = \alpha^r \times T_0$

```
1   IF[1] FIRST PHASE
2       l = randBetween(1, L)
3       l' = randBetween(1, L)
4       IF[2] [l ≠ l' and β1 > rand()]
5           FOR[1] t = 1 to t = T
6               m = X[t][LHS-segment][l][M_{l,t}][a]
7               m' = X[t][LHS-segment][l'][M_{l',t}]
8
9               X[t][LHS-segment][l][M_{l,t}] = m'
10              X[t][LHS-segment][l'][M_{l',t}] = m
11          END FOR[1]
12      END IF[2]
13  END IF[1]
15 .....
15  IF[3] SECOND PHASE
16      FOR[3] t = 1 to t = T
17          l = randBetween(1, L)
18          l' = randBetween(1, L)
19          IF[4] [l ≠ l' and β1 > rand()]
20              m = X[t][LHS-segment][l][M_{l,t}]
21              m' = X[t][LHS-segment][l'][M_{l',t}]
22
23              X[t][LHS-segment][l][M_{l,t}] = m'
24              X[t][LHS-segment][l'][M_{l',t}] = m
25          END IF[4]
26      END FOR[3]
27  END IF[3]
```

[a] $X[t]$[LHS-segment]$[l][M_{l,t}]$ means a solution X, segment t at level-I, LHS-segment at Level-II, location l, machine index $M_{l,t}$ (see Figure 4.2)

Figure 4.6: Pseudocode for Machines Locations Swap Operator (MLSO)

```
1   IF[1] [β2 > rand()]
2       t1 = randBetween(1, T)
3       t2 = randBetween(t1 + 1, T)
4       l = randBetween(1, L)
5       l* = randBetween(1, L)
6
7       m* = X[t1][LHS-segment][l][Ml,t][a]
8       m' = X[t1][LHS-segment][l*][Ml*,t]
9
10      X[t1][LHS-segment][l][Ml,t] = m'
11      X[t1][LHS-segment][l*][Ml*,t] = m*
12
13      FOR[2] t = t1 + 1 to t = t2
14          FOR[3] l = 1 to L
15              IF[2] X[t][LHS-segment][l][Ml,t] = m*
16                  BREAK FOR[3] (carry the value of l)
17              END IF[2]
18          END FOR[3] 19
20          m' = X[t][LHS-segment][l][Ml*,t]
21          X[t][LHS-segment][l][Ml,t] = m'
22          X[t][LHS-segment][l*][Ml*,t] = m*
23      END FOR[2]
24  END IF[1]
```

[a] $X[t][\text{LHS-segment}][l][M_{l,t}]$ means a solution X, segment t at level-I, LHS-segment at Level-II, location l, machine index $M_{l,t}$ (see Figure 4.2)

Figure 4.7: Pseudocode for Machine Locate and Fix Operator (MLFO)

Q	Number of iterations to be performed in each search path at each temperature level
F	Interaction frequency. It is defined as the number of iterations to be performed by search path before interaction is effected among these search paths.
BI	Best feasible individual so far found
$Phase$	The current phase of the search which equals to 1 for the first phase or 2 for the second phase
C_{phase}	The number of iterations to be performed by each search path before the search phase is changed from $Phase = 1$ to $Phase = 2$

4.3. Pure Simulated Annealing (PSA)

Although LPSA is efficient search and allows creating tours of high quality, solving a large LP-subproblem is memory intensive in CPLEX. Insufficient physical memory is one of the most common problems when running large LPs in CPLEX, similar to the large-size problems proposed to solve in numerical examples in Chapter 5. When CPLEX recognizes that a limited amount of memory is available it automatically makes algorithmic adjustments to compensate. These adjustments almost always reduce optimization speed. One remedy is to consider a heuristic solution procedure for LP-subproblem, but both heuristically and optimally solution approaches suffer from trade-off between computation time and solution quality. The heuristic solution of the LP-subproblem may be done quite fast, but the quality of solution is expected to be superior in case of optimally solved LP-subproblems. In order to examine and demonstrate the capability of LPSA to handle large-scale problems, we also developed a Pure Simulated Annealing (PSA) algorithm presented in Sections 4.3.2 to 4.3.6 .

In contrast to LPSA, PSA searches over both discrete and continuous variables

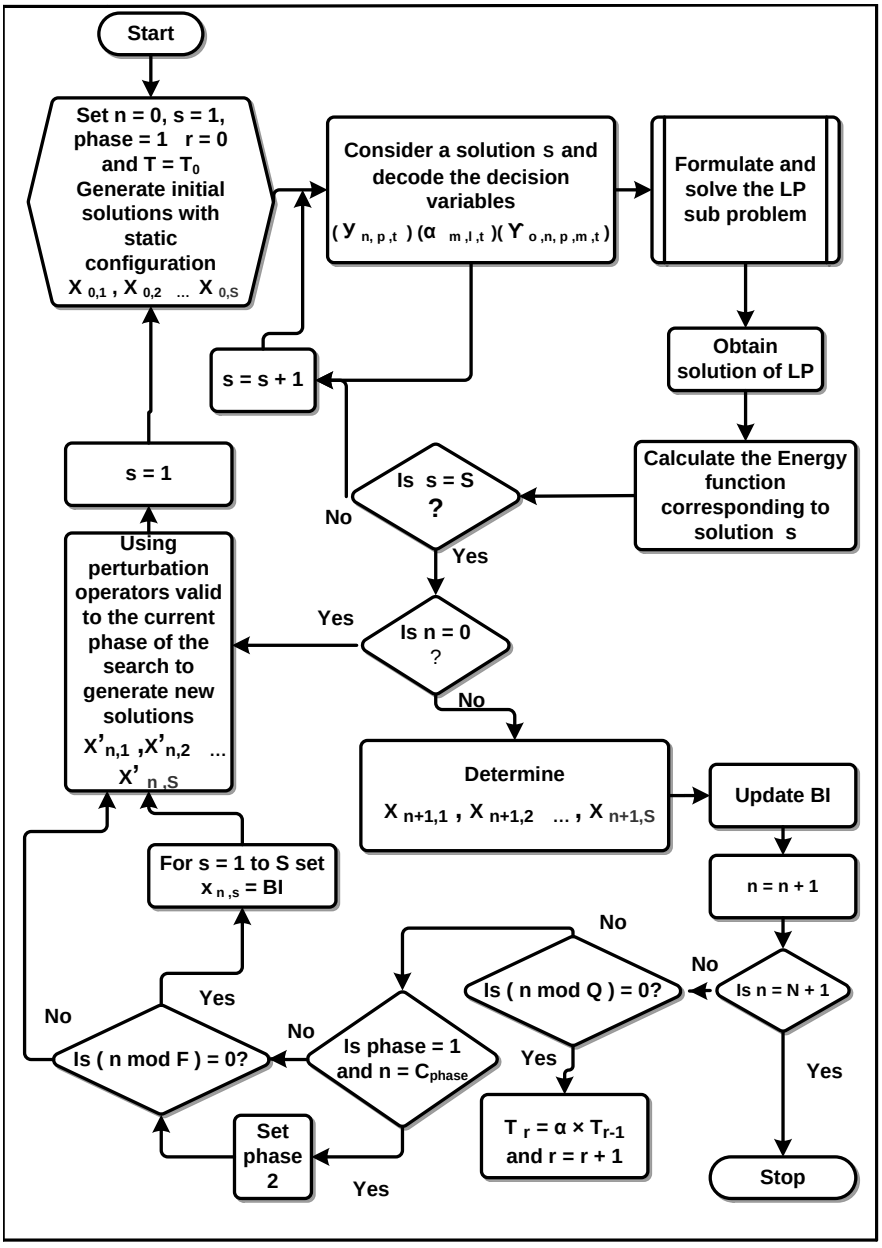


Figure 4.8: The steps of LPSA

in the solution space. Therefore, the problem space is drastically increased and it implies that employing a larger number of search paths in the algorithm can allow it to exhaustively search relatively large promising regions of the solution space.

4.3.1. Solution Representation in PSA

To explore the search space described above, a solution encoding is developed that can be decoded to give a valid combination of the values of the integer and continuous variables. This solution encoding scheme is given in Figure 4.9. All decision variables in solution representation in LPSA are totally part of the PSA encoding scheme and decoded in the same manner. The only difference is the inclusion of variables $y_{p,t}$, $S_{p,t}$, $R_{p,t}$, and $X_{n,p,t}$. Two variables $y_{p,t}$ and $S_{p,t}$ take value 0 or 1 to represent the choice of in-house production and subcontracting production for each product in each time period respectively. Subcontracting ratio $R_{p,t}$ takes a random value in interval [0,1] to account for the proposition of the total demand of part p subcontracted during period t. The $X_{n,p,t}$ assume values in [0, 1] and are used to calculate the size of the n^{th} sublot of product p in time period t ($b_{n,p,t}$).

In course of search, it is possible for all sublots of a product to have a size of zero where their corresponding $X_{n,p,t}$ have a value equal to zero. In such case, in order to compute the size of each sublot for that product, the production lot size of that product in that time period ($v_{p,t}$) is assumed to be equally divided between the maximum number of sublots (Np), which is known for each product in each period. In addition, if all no sublot are created for a production of a product p in a period t (i.e. all $y_{n,1,1} = 0$) and in-house production for that product in that time period is read to be one (i.e. $y_{1,1} = 1$), one of the sublots among maximum number of sublots for that product is randomly chosen to arbitrarily set at 1. This enforces that at least one sublot needs to be created to process $y_{1,1}$ if it is

required to be processed. By decoding a solution point under consideration, the corresponding continuous variables $v_{p,t}$, $\hat{v}_{p,t}$, $h_{p,t}$ are determined using the Lot Sizing Heuristic presented in Section 4.3.3.

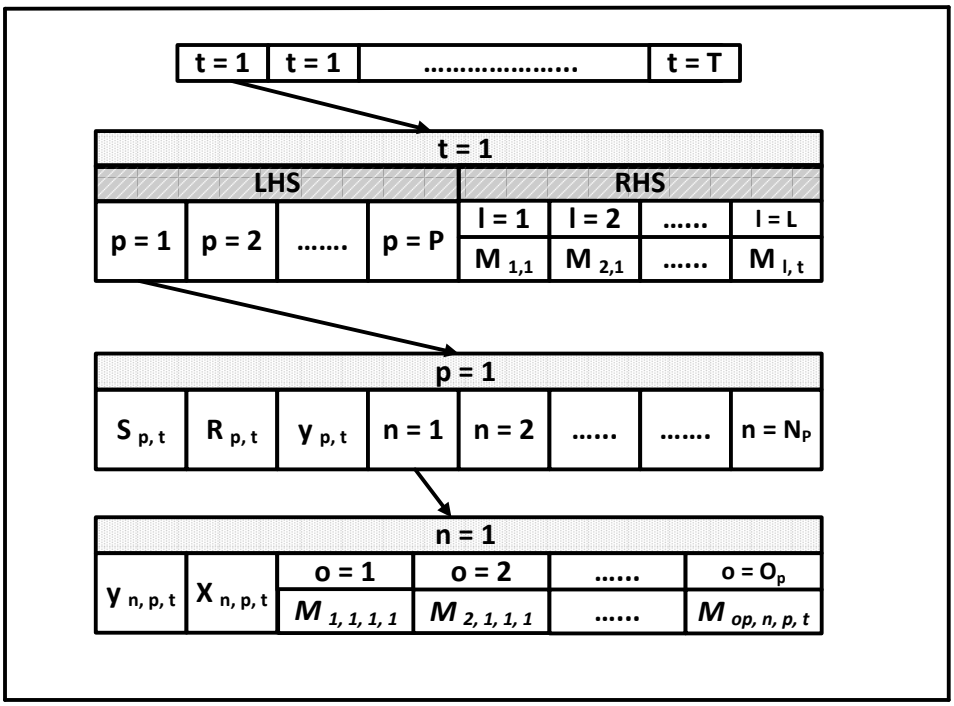


Figure 4.9: Solution Representation in PSA

4.3.2. Decoding Solution Representation

Similar to LPSA, the values of decision variables $\alpha_{m,l,t}$ and $\gamma_{o,n,p,m,t}$ are determined using Eq. (4.2) and Eq. (4.3) ,and $M_{l,t}$ along with $M_{o,n,p,m,t}$ can be directly read from the solution representation. The set of production planning decision variables including $v_{p,t}$, $\hat{v}_{p,t}$ and $h_{p,t}$ are computed by Lot Sizing Heuristic explained in the next Section. The value of each $X_{n,p,t}$ is read from the solution encoding scheme and used to decode $b_{n,p,t}$ using Eq. (4.7) where $v_{p,t}$ is known from previously performed Lot Sizing Heuristic.

$$b_{n,p,t} = \begin{cases} \frac{X_{n,p,t}}{\sum_{n=1}^{N_p} X_{n,p,t}} \times v_{p,t} & ; \text{ If } \sum_{n=1}^{N_p} X_{n,p,t} > 0 \\ \\ \frac{v_{p,t}}{N_p} & ; \text{ Otherwise} \end{cases} \tag{4.7}$$

The value of $y_{n,p,t}$ can be directly read from solution representation. Given the value of $\alpha_{m,l,t}$ the variable $e_{m,t}$ can be determined such that $e_{m,t} = E_{l,l'}$ if the variables $\alpha_{m,l,t-1} = \alpha_{m,l,t} = 1$.

Recall that, the constraints in Eqs. (3.4) and (3.5) of the main model are used to set $d_{o,n,p,t} = E_{l,l'} \cdot b_{n,p,t}$ if the sum of the binary decision variables $\alpha_{m,l,t}$, $\gamma_{o,n,p,m,t}$, $\alpha_{m',l',t}$, and $\gamma_{o+1,n,p,m',t}$ is equal to four. Knowing the values of these four binary variables, $d_{o,n,p,t}$ is computed as shown in Eq. (4.8). The last variable $\delta_{o,n,p,m,t}$ is also can be computed by Eq. (4.9). This equation is required to give $\delta_{o,n,p,m,t} = 0$ if the binary variable $\gamma_{o,n,p,m,t} = 0$.

$$\begin{gathered} d_{o,n,p,t} = E_{l,l'} \cdot b_{n,p,t} \ ; \\ \forall (o,n,p,t,m,l,m',l') | \, (\alpha_{m,l,t} + \gamma_{o,n,p,m,t} + \alpha_{m',l',t} + \gamma_{o+1,n,p,m',t} = 4) \end{gathered} \tag{4.8}$$

$$\delta_{o,n,p,m,t} = U_{o,p} \cdot b_{n,p,t} \ ; \quad \forall (o,n,p,m,t) | \gamma_{o,n,p,m,t} = 1 \tag{4.9}$$

4.3.3. Lot Sizing Heuristic

As it has been stated previously, the variables $R_{p,t}$, $S_{p,t}$, and $y_{p,t}$ can be used to heuristically compute three continuous variables $v_{p,t}$, $\hat{v}_{p,t}$ and $h_{p,t}$ in each solution point. Lot Sizing heuristic was applied for each solution point visited during the search so that the corresponding value of the energy function can be evaluated. This heuristic is based on two important properties which are common in lot sizing problems in the literature: (1) whenever a setup is introduced for an item (i.e. $y_{1,t_1} = 1$), we assume that enough quantity for this item is produced to satisfy

the demands of any periods until the period immediately before the next setup for this item (i.e. $y_{1,t_2} = 1$), and (2) Any subcontracting lot size is not necessarily required to exactly perform at period when $S_{p,t} = 1$ but it can be processed at period when demand needs it to satisfy itself. By doing this, inventory holding is decreased and there is no need to outsource products in advance. We can determine the $v_{p,t}, \hat{v}_{p,t}$ and $h_{p,t}$ from decoded $R_{p,t}$, $S_{p,t}, y_{p,t}$ and given $D_{p,t}$ in each solution point using following steps 1 to 5.

1. For any product p and any period t, if $y_{p,t} = 0$, then $v_{p,t} = 0$.

2. For any product p and any periods $t_1 < t_2 \leq T$ if $y_{p,t_1} = y_{p,t_2} = 1$ and $y_{p,t} = 0$ for all $t_1 < t < t_2$, then

$$v_{p,t_1} = \begin{cases} \sum_{t=t_1}^{t_2-1} Dp,t \times (1 - R_{p,t_1}) & ; \text{ If } S_{p,t_1} = 1 \\ \\ \sum_{t=t_1}^{t_2-1} Dp,t & ; \text{ Otherwise} \end{cases} \tag{4.10}$$

3. For any product p and any period t

$$h_{p,t} = \begin{cases} v_{p,t} + \hat{v}_{p,t} - D_{p,t} & ; \text{ If } t = 0 \\ \\ v_{p,t} + \hat{v}_{p,t} + h_{p,t-1} - D_{p,t} & ; \text{ If } t > 0 \end{cases} \tag{4.11}$$

4. For any product p and period $t = 1$

$$\hat{v}_{p,1} = \begin{cases} D_{p,1} - v_{p,1} & ; \text{ If } v_{p,1} < D_{p,1} \\ \\ 0 & ; \text{ Otherwise} \end{cases} \tag{4.12}$$

5. For any product p and any period $t > 1$

$$\hat{v}_{p,t} = \begin{cases} D_{p,t} - v_{p,t} - h_{p,t-1} & ; \text{ If } v_{p,t} - h_{p,t-1} < D_{p,t} \\ 0 & ; \text{ Otherwise} \end{cases} \tag{4.13}$$

For example, assume we have demand data (Dp, t) and decoded binary variables ($y_{p,t}, R_{p,t}$ and $S_{p,t}$) of a solution point for a problem consisting of only one product producing in eight periods as given in Table ??. The Lot Sizing heuristic can be used to obtain continuous variables associated with in-house production, subcontracting and inventory holding in every period as shown in Tale ??.

	Periods							
	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
$y_{p,t}$	1	0	0	1	0	0	0	1
Dp, t	100	80	20	50	150	50	200	100
$\sum_{t=t_1}^{t_2-1} Dp, t$	200	0	0	400	0	0	0	100
$v_{p,t}$	180	0	0	360	0	0	0	80
$h_{p,t}$	80	0	0	310	160	110	0	0
$\hat{v}_{p,t}$	0	0	20	0	0	0	90	20
$R_{p,t}$	0.1	0.05	0.8	0.1	0.7	0.06	0.3	0.2
$S_{p,t}$	1	0	0	1	0	0	0	1

Table 4.1: Example problem data and calculated variables using Lot Sizing heuristic

4.3.4. Constraints Handling

Traditional SA algorithms serve as search techniques. However, due to the use of perturbation operators to manipulate the solution point in each iteration, SA may

produce infeasible solutions when used to solve constraint optimization problems. Infeasibility stem from the fact that the created solutions violate some constraints of the original problem. Different approaches to handle constraints, such as rejection of infeasible solution, penalty methods or repair algorithms are proposed in the literature. In this study, the penalty technique was used to handle constraints imposing workload balancing and capacity requirements. A penalty method adds penalty quantities to objective function if the corresponding constraints are not satisfied. On the other hand, some constraints such as those in Eq. (3.18) and Eq. (3.19) are handled using inherent feature in structure of solution representation explained in Section 4.3.2. The Lot Sizing heuristic handles constraints in equations Eq. (3.6), Eq. (3.7) and Eq. (3.8). This heuristic is applied in each iteration of PSA to determine the continuous variables.

Recall that in our mathematical model, Eq. (3.14) permits workload balancing. We propose this constraint to be relaxed and incorporated into the objective function through a penalty term. Eq. (4.14) and Eq. (4.15) are used to calculate minimum allowable and actual work performed. As mentioned in Section 3.4, we assume a workload which is using a particular RE need to be evenly divided among the machines overlapping this RE if the the workload factor Υ is set to one. Eq. (4.15) computes allowable workload for each machine m sharing a resource element r among the other machines in each time period. The actual workload for a machine in a time period is given using Eq. (4.15) which apparently is numerator of the Eq. (4.14). The factors c_{wb} is used to scale the penalty terms corresponding to these constraints.

Another constraint chosen to be handled by a penalty method is capacity requirements. Eq. (4.16) guarantees that the workload on machine m in time period t is less or equal to the available time C. The second term in Eq. (4.16) is used to calculate assigned workload to a machine at a particular time period. If this assigned workload is larger than the total available capacity for this machine

at that particular period infeasibility occurs and the negative value of $SC_{m,t}$ is added to the objective function.

$$WL_{min}(r,m,t) = \frac{\sum_{m''=1}^{M}\sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{o=1}^{O_p} B_{r,o,p} \times \delta_{o,n,p,m'',t}}{\sum_{m'=1}^{M} A_{r,m'}} \times \Upsilon; \forall(r,m,t) \tag{4.14}$$

$$WL_{actual}(r,m,t) = \sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{o=1}^{O_p} B_{r,o,p} \times \delta_{o,n,p,m,t}; \forall(r,m,t) \tag{4.15}$$

$$SC_{m,t} = C_t - \sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{o=1}^{O_p} \delta_{o,n,p,m,t}; \forall(m,t) \tag{4.16}$$

4.3.5. Fitness Evaluation

A candidate solution point in search space is evaluated according to a fitness function that is usually synonymous with the objective function of the model. However, if a constraint (s) is violated, a penalty function term can be added to the objective function to relax some proposed constraints. In PSA algorithm, workload balancing and capacity violation penalty terms are incorporated in objective function as shown in Eq. (4.17).

$$\begin{aligned} F.F = \quad & \text{Model Objective Function} \\ + \quad & c_{wb} \cdot \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{r=1}^{R} \max\{0, \quad WL_{min}(r,m,t) - WL_{actual}(r,m,t)\} \\ + \quad & SC_{m,t} \end{aligned} \tag{4.17}$$

4.3.6. Perturbation Operators

A promising algorithm requires well designed multiple perturbation operators to create new solutions from a given current solution in order to adequately meet local neighborhood structure requirements. In the PSA, we developed seven different solution perturbation operators. These are: (1) Machine Locations Swap

Perturbation Operator (MLSPO), (2) Production Flip Operator (PFO), (3) Alternative Machine Perturbation Operator (AMPO), (4) Outsourcing Flip Operator (OFO), (5) Subcontracting Ratio Perturbation Operator (SRPO), (6) Sublot Flip Operator (SFO) and (7) Sublot Size Determination Perturbation Operator (SSDPO). To avoid a random search, all perturbation operators are applied with small probability.

Similar to the LPSA, the search process in the proposed PSA is divided into two phases such that the algorithm aims to find the best static machine configuration in the first phase. In contrast, in the second phase the algorithm seeks to find an optimal reconfiguration based on the solution found in the first phase. In the first phase of the search, MLSPO which is identical to the location swap operator proposed in the LPSA is applied. In the dynamic phase, the MLSO in the LPSA algorithm is analogously applied and renamed MLSPO. We omit the details as any interested reader can proceed as in Section 4.2.5. The PFO is applied to each of the $y_{p,t}$ to switch its value between 0 and 1. The operator AMPO is applied on each $M_{o_p,n,p,t}$ in the LHS-segments of a solution to alter its value to one of machines that can process operation o of product p. SRPO and SSDPO are used to make a few alterations to the value $R_{p,t}$ and $X_{n,p,t}$ respectively. These operators randomly step down and up the value of these variables within [0,1] by a certain step value(θ). Pseudocode for applying SRPO with probability of λ is shown in Figure 4.10. The SSDPO scheme is conceptually same with SRPO.

The OFO and SFO flip the value of binary variables $S_{p,t}$ and $y_{n,p,t}$ respectively. SFO may make $y_{n,p,t}$ equal to zero while $y_{p,t}$ is 1 and requires at least one sublot being created. To avoid this situation and to end up with a properly lot streaming, one sublot among maximum number of sublot for a particular p (N_p) is randomly selected and set to one. The mechanism is illustrated in Pseudocode given in Figure 4.11.

```
1  FOR[1]  t = 1 to t = T
2     FOR[2]  p = 1 to p = P

3        IF λ > rand()

4           θ[a] = Maxθ [b] ×rand()

5           IF rand() < 0.5

6           X'[t].LHS[p].Rpt = X[t].LHS[p].Rpt [c] + θ

7           ELSE

8           X[t].LHS[p].Rpt = X[t].LHS[p].Rpt − θ

9     END FOR[2]
10 END FOR[1]
```

[a] The θ parameter is used to step up and step down the subcontracting ratio.
[b] $Max\theta$ is between (0, 1) and given as data.
[c] $X[t].LHS[p].Rpt$ means trial solution X', segment t at level-I, RHS-segment at Level-II, product p, subcontracting ratio $R_{p,t}$ (see Figure 4.9)

Figure 4.10: Pseudocode for applying Subcontracting Ratio Perturbation Operator

4.4. Linear Programming Embedded Genetic algorithms

Following sections describe linear programming embedded genetic algorithm (LPGA) solving the mathematical model problem presented in Chapter 3. LPGA utilizes a Simplex method to solve the linear programming sub problem corresponding to a given solution point. In similar manner to LPSA proposed in Chapter 4 the multiple search path LPGA searches over discrete variables in the solution space and corresponding continuous variable are determined by solving LP-subproblem.

Genetic algorithms are superior to simulated annealing algorithms in covering a much larger landscape of the search space at each iteration because GA uses

```
1   FOR[1]  t = 1 to t = T
2     FOR[2]  p = 1 to p = P
3       FOR[3]  n = 1 to n = N_p
4         IF β > rand()

5             IF X'[t].LHS[p].Sub[n].Ynpt = 1

6             X'[t].LHS[p].Sub[n].Ynpt = 0
7         ELSE

8             X'[t].LHS[p].Sub[n].Ynpt = 1

9       END FOR[3]

10        FOR[4]  n = 1 to n = N_p

11        Some-Ynpt[a] = Σ_{n=1}^{N_p} X'[t].LHS[p].Sub[n].Ynpt

12        END FOR[4]

13        IF Σ_{n=1}^{N_p} X'[t].LHS[p].Sub[n].Ynpt = 1 ∧ Some − Ynpt = 0

14          X'[t].LHS[p].Sub[randBetween (0, N[p] − 1)].Ynpt = 1

15    END FOR[2]
16  END FOR[1]
```

[a]The parameter Some -Ynpt is used to detect a particular situation where all Ynpt are zero.

Figure 4.11: Pseudocode for applying Sublot Flip Operator

a population based selection while SA utilizes one point in each iteration. In addition, recombination operators in GAs enable them to mix good characteristics from different solution to find solution with better quality whereas SA does not gain much of this. However, SA is well-known for its simple implementation and so often faster than GA. In Chapter 5, we use same numerical examples for the proposed model to investigate computational time and solution quality resulting from both LPSA and LPGA.

4.4.1. Genetic Algorithm

The Genetic algorithm invented by Holland (in year 1975) is stochastic search techniques based on population-based solution search according to the principles of the natural evolution process. GA was discovered as a useful tool for a wide range of combinatorial optimization problems. The genetic process starts by generating an initial population of chromosomes (solutions) and evolving this population over time until near optimal solution is obtained. Using reproduction operators over solutions of the problem, new generations are evolved by breeding the pairs of existing chromosomes. The offspring, just by combining all the good features from its parents, may outperform its parents. Both offspring and parent chromosomes are then assessed against the fitness function. Finally, individuals with the highest fitness level will survive to form the next generation. Figure 4.12 shows how a genetic algorithm loops over an iteration process to make the population evolve.

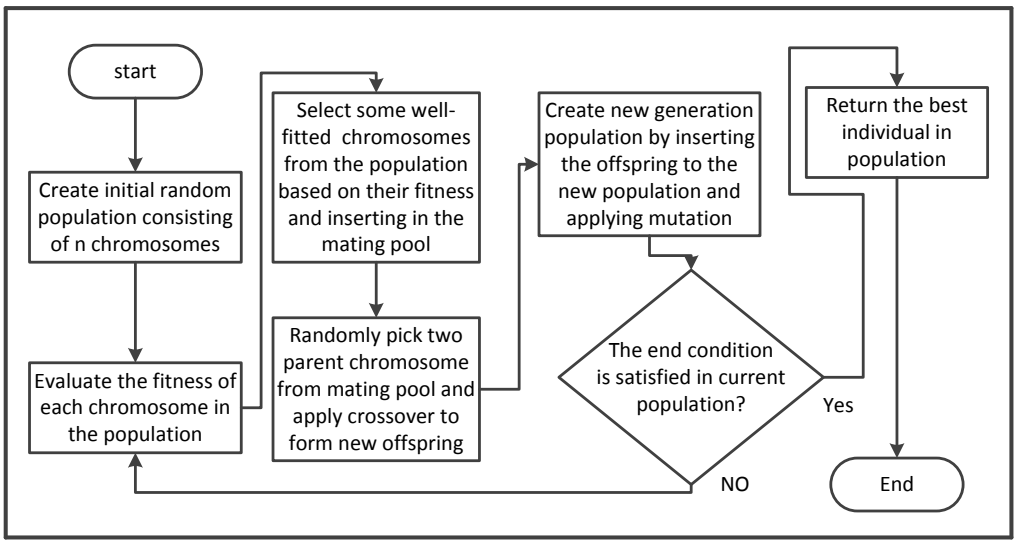


Figure 4.12: Executional steps of a genetic algorithm

4.4.2. LP Embedded Genetic Algorithm

Some parts of LPGA are identical to ones considered in LPSA discussed in Chapter 4. Solution representation, the decoding methods of integer variables and the formulating LP-subproblem are parts of both algorithms which are totally similar. In order to avoid repeated explanation, the preference is given to recombination operators and fitness function used in LPGA in the following sections.

4.4.3. Fitness Function

In the LPGA, the potential solution represented by each chromosome in the population of candidate solutions is evaluated against a fitness function that is constructed based on the objective and constraint functions of the model. For a given solution, its fitness is given by Eq. (4.18) which is identical to the objective function of the model (Eq. (3.1)).

$$\begin{aligned} Minimize(E) \;=\; & \sum_{t=2}^{T}\sum_{m=1}^{M}(G_m \cdot e_{m,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{n=1}^{N_p}\sum_{o=1}^{O_p-1}(F_p \cdot d_{o,n,p,t}) \\ + \; & \sum_{t=1}^{T}\sum_{p=1}^{P}(H_p \cdot h_{p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{n=1}^{N_p}(S_p \cdot y_{n,p,t}) \\ + \; & \sum_{t=1}^{T}\sum_{p=1}^{P}(\Theta_p \cdot v_{p,t}) + \sum_{t=1}^{T}\sum_{p=1}^{P}(\hat{\Theta}_p \cdot \hat{v}_{p,t}) \end{aligned} \tag{4.18}$$

It should be noted that GAs deal with maximization problems, thus, if the problem is modeled as minimization, the cost function can be easily transformed into a fitness function by inverting it. This is obtained using Eq. (4.19) where E_{max} and E_{min} are the maximum and the minimum values of the raw fitness E in the current population.

$$\widetilde{E} = \begin{cases} 1 & ; \ if \ E_{max} = E_{min} \\ \frac{E_{max}-E}{E_{max}-E_{min}} & ; \ if \ \frac{E_{max}-E}{E_{max}-E_{min}} > 0.1 \\ 0.1 & ; \ otherwise. \end{cases} \tag{4.19}$$

4.4.4. Selection Operator

Selection process is defined as selecting of two parents from the population for recombination. Using a fitness function, each chromosome is evaluated and highly fitted chromosomes selected for mating in hopes that their resulting offspring are fitter individuals (Sivanandam and Deepa, 2007). The genetic algorithm is properly directed toward promising regions in the search space by selection process. In tournament selection (Goldberg and Holland, 1988), a set of chromosomes is randomly chosen from the population and the one with highest fitness is selected for reproduction. The number of chromosomes in the set is called tournament size. The set selection process and tournaments are repeated until a desired size of mating pool has been formed. The winners may be drawn from the population with or without replacement. This method is widely used in GA applications due to its efficiency and coding simplicity. This selection method has been employed in this work and pseudocode to perform a tournament with pre-specified size to select a chromosome for the population of known size is shown in Figure 4.13.

4.4.5. Crossover Operators

Crossover operators are applied directly on the pairs of parent chromosomes identified from the selection step to give birth one or more offspring. Crossover operator is performed on parents in a mating pool with the hope that it reproduces offspring with relatively better fitness levels (Sivanandam and Deepa, 2007)). By

```
1   FOR[1]   p = 1 to p = toursize[a]
2
3            tour[p] = randBetween(1, popsize)[b]
4
5   END FOR[1]
6   FOR[2]   p = 1 to p = toursize
7
8       IF[1]   objective(parent) > objective(tour[p])
9               parent = tour[p]
10
11      END IF[1]
12  END FOR[2]
13  return parent
```

[a] Tournament size
[b] Population size

Figure 4.13: A Pseudocode to perform a tournament to select a chromosome for a population.

doing this, the new population is expected to be enriched with the better chromosomes. In addition to appropriately designed crossover operators that are tailored to the structure of the solution representation, setting up a crossover rate, or probability, is crucial in GA application. The crossover probability is the parameter in GA algorithms that determine the probability at which a crossover operator is applied. A higher crossover probability inserts new strings more quickly into the population. Too high crossover probability causes high-performance chromosomes to be eliminated faster than selection can produce improvements. A low crossover rate may experience stagnation because of the lower exploration (Sundhararajan and Pahwa, 1994). In this Section, we present two crossover operators. These are All-information Period Swap Crossover (APSC) and All-period Location Swap Crossover (ALSC). The first crossover randomly chooses a period in the planning horizon and swaps all genetic information corresponding to that period. The mechanism of this crossover operator is illustrated in Figure 4.15. All-period Location Swap Crossover operator is a two-point crossover to exchange layout configuration sub-segment between the parents in all periods, in contrast

to PLSC which is applied only on a randomly selected period.

4.4.6. Mutation Operators

Unlike the crossover operators, mutation operators are applied on each chromosome to reverse a selected chromosome bit pattern. By this concept, lost or disturbing genetic information is recovered. In contrast to crossovers aim to exploit the current solution to create better fitted ones, mutations are considered to assist whole solution space exploration. Another important role of mutation is to escape from being trapped in local minima (Sivanandam and Deepa, 2007). Similar to crossovers, mutations are only applied on a selected chromosome based on pre-specified probability. Too low mutation rate results in higher chances of being trapped in local optima and in too high mutation rate there would be too many random perturbations and offspring might lose their resemblance to the parents. In this Section, we describe four mutation operators used in the LPGA algorithm. These are: (1) Machines Locations Swap Mutator (MLSM), (2) Machine Locate and Fix Mutator (MLFM), (3) Sublot Mutator, and (4) Alternative Machine Swap Mutator (AMSM). The MLSM is applied with small probability in two different manners depending on the phase of the search. In the first phase of the search, the operator first arbitrarily selects two loci l and l'. Then, for these pair of locations, it swaps the values of $M_{l,t}$ and $M_{l',t}$ of the Sub-segment of each period of the chromosome. Applied in this manner, the operator keeps the sub-segments labeled locations identical across the periods and results in a trial solution with static configuration. Likewise in doing such swaps, in the second phase, MLSM selects the pair of locations l and l' for each period independently resulting in a non-identical sub-segments of the affected solution. The second layout mutation, MLFM, is applied only in the second phase of the search. It fixes an arbitrarily chosen machine (m') at an arbitrarily chosen location (l') between arbitrarily chosen time periods t_1 and t_2 (such that $t_1 < t_2 \leq T$). The essence of

this mutation is that once a particular machine is moved to a particular location, it is more likely for this machine to remain in that particular location for a certain number of time periods. The Sublot Mutator is applied to each of the $y_{n,p,t}$ in the chromosome to switch its value between 0 and 1. The AMSM randomly alters the $M_{o,n,p,t}$'s value to one of machines that can process operation o of product p. Typical results of this mutator are shown in Figure 4.14.

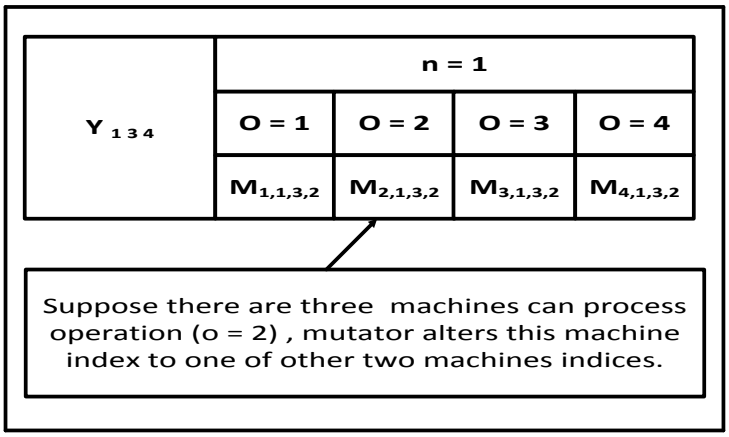


Figure 4.14: An example of Alternative Machine Swap Mutator applied to second operation of the first sublot of p = 3 in time period t = 4

4.4.7. Steps of the LPGA

The general steps of LPGA combine two genetic and linear programming algorithms such that the LP-subproblem corresponding to each individual in the population in each generation is iteratively solved. This is described by means of a flowchart given in Figure 4.16.

N	Population size
c	Index for a chromosome in a given population
k	Generation counter

```
PeriodXover(Parent[1], Parent[2] // Period crossover for parents 1 and 2

1   t* = randBetween(1,  T)

2   IF [1] (random() < PeriodXoverPro [a])
3       FOR[1]   m = 1 to m = M                    // Machine location swap.

4               LocationIndex = Pare[1].Peri[t*].MachLoca[m] [b]
5               Pare[1].Peri[t*].MachLoca[m]= Pare[2].Peri[t*].MachLoca[m]
6               Pare[2].Peri[t*].MachLoca[m]= Location index

7       END FOR[1]
8       FOR[2]   p = 1 to p = P                    // sublots swap
9           FOR[3]   n = 1 to n = Np
10              SublotValue = Pare[1].Peri[t*].Prod[p].Sublot[n].Ynpt [c]
11              Pare[1].Peri[t*].Prod[p].Sublot[n].Ynpt = Pare[2].Peri[t*].Prod[p].Sublot[n].Ynpt
12              Pare[2].Peri[t*].Prod[p].Sublot[n].Ynpt = SublotValue

13          END FOR[3]
14       END FOR[2]
15       FOR[4]   p = 1 to p = P                    // Machine performing operation swap
16          FOR[5]   n = 1 to n = Np
17              FOR[6]   o = 1 to o = Op

18              MachIndex = Pare[1].Peri[t*].Prod[p].Sublot[n].Machi[o] [d]
19              Pare[1].Peri[t*].Prod[p].Sublot[n].Machi[o] = Pare[2].Peri[t*].Prod[p].Sublot[n].Machi[o]
20              Pare[2].Peri[t*].Prod[p].Sublot[n].Machi[o]= MachIndex

21      END FOR[6]
22          END FOR[5]
23              END FOR[4]
24  END IF[1]
```

[a] A probability of applying crossover

[b] means machine m located at location "LocationIndex" in time period t^* in parent solution 1.

[c] means sublot n of product p in time period t^* in parent solution 1 has value equal to "SublotValue".

[d] means a attractive machine with "AltrMachIndex" index is selected to perform operation o of sublot n of product p in time period t^* in parent solution 1.

Figure 4.15: Pseudocode for applying All-information Period Swap Crossover in LPGA.

$MaxGen$	Maximum generations
$Phase$	An indicator number which equals to 1 for the static phase or 2 for the dynamic phase
g_{phase}	Generation at which the value of $Phase$ should be set equal to 2 if it were not previously set to this value by other conditions
WoI	Number of successive population rejuvenations counted without any improvement of the best individual so far found
WoI_{max1}	Maximum value of WoI at which point the second phase is to be entered if $Phase$ was equal to 1
WoI_{max2}	Maximum value of WoI in the second phase at which point the search will be terminated.

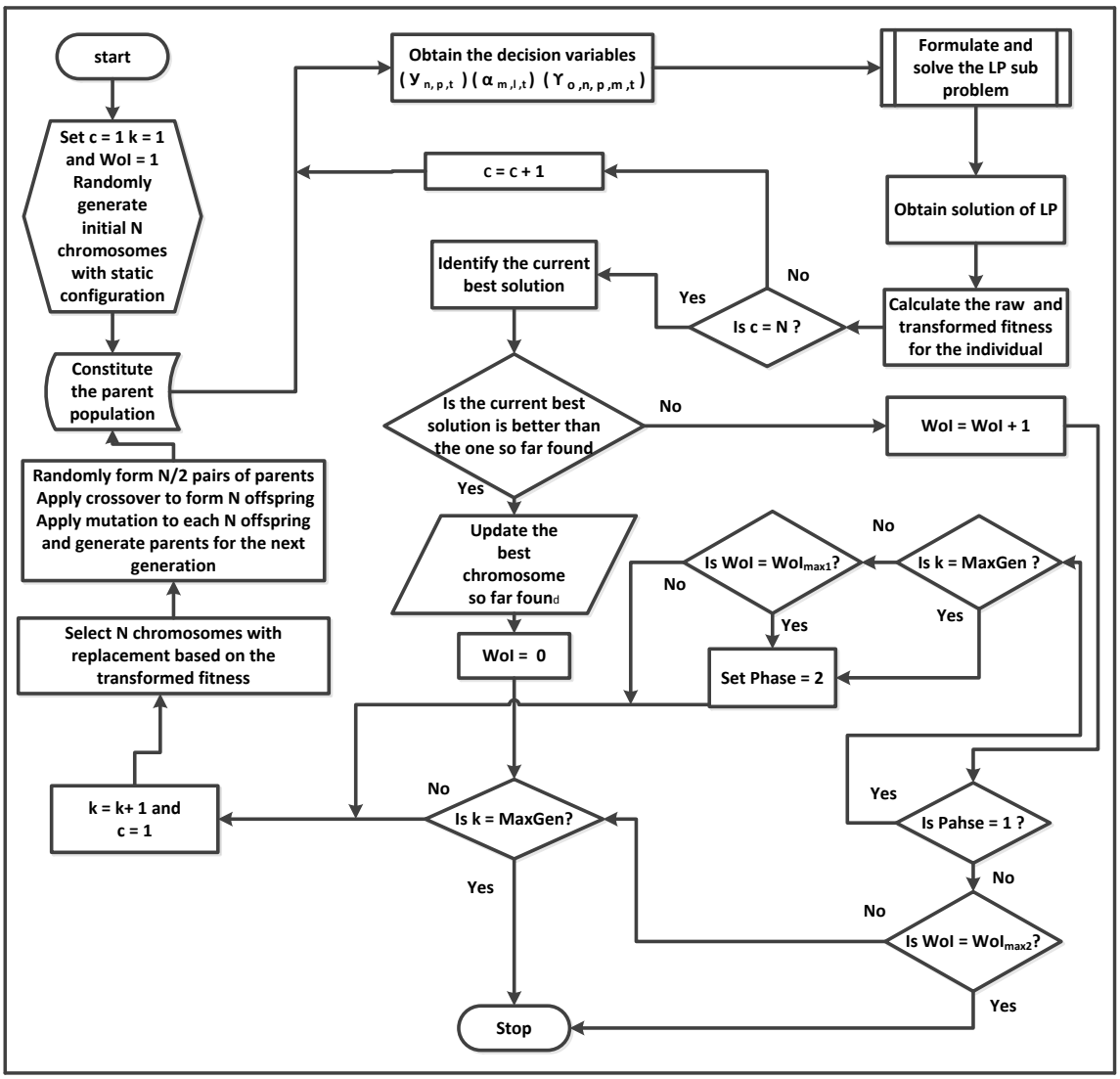


Figure 4.16: The steps of LPEGA

Chapter 5

Numerical Examples: Distributed Layout Design

5.1. Model Analysis

Since the comprehensive problem addressed in this thesis has not been previously presented, we have no comparable examples from the literature to use. Therefore, we generated several data sets to illustrate the problem and demonstrate the performance of the proposed solution procedure. One of these data sets (referred to as Problem 1) is provided in detail in Appendix **??**. In this data set, we considered a system composed of 20 resource elements and 22 machine tools. Table A.1 shows four different cases in which each of 20 REs is available on one or more machines. More specifically, Case 1 represents a situation in which a particular RE is available on several machine tools; Case 4 represents a situation where most of the machines have unique capabilities; and Cases 2 and 3 lie in between the two extremes. The average number of machines per RE in these four cases is 4.55, 2.65, 1.5, and 1.1, respectively. In Table A.2 are the model parameters (Θ_p, $\hat{\Theta}_p$, H_p, F_p, S_p, N_p, O_p), the index of the required resource element r for each operation, and the processing time $U_{o,p}$. The demands for the parts in

four planning periods are provided in Table A.3. The relocation cost G_m for each machine type m is in Table A.4.

The layout showing potential machine locations in Problem 1 is provided in Figure A.1. Although the proposed model can address any type of layout shape and material handling system, we prefer to adopt a system served by automated guided vehicles (AGVs) arranged in tandem configuration. AGVs are preferable to stationary material handling robots because of their mobility, and to conveyors because of their flexibility (Asef-Vaziri and Laporte, 2005). An AGV system can be reconfigured to accommodate changes in production volume, product mix, product routing, and equipment interface requirements more readily than most other material handling systems (Goetz and Egbelu, 1990). In Table A.5, we provide the locations of machines in an arbitrarily generated functional layout (where similar machines are placed in close proximity) and five arbitrarily generated distributed layouts (DL1, ..., DL5). The material handling and machine relocation distances between each pair of locations are shown in Tables A.6 and A.7, respectively. In Problems 2 to 6, we considered the processing of 35, 50, 65, 80, and 120 parts, respectively. The maximum number of operations per part was six (in Problems 2, 3 and 4) and eight (in Problems 5 and 6). The problems were solved using the proposed algorithm. The algorithm was implemented in C++ and interfaced with ILOG CPLEX version 12.2 to solve the linear programming subproblem.

5.1.1. Functional versus Distributed Layout

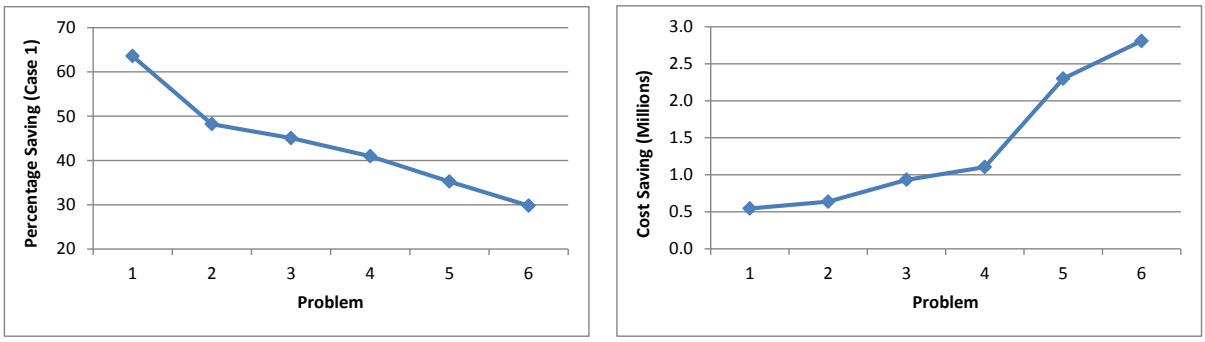
The aim of this Section is to illustrate the greater effectiveness of using distributed layouts compared to using a functional layout in a situation where there are machine tools with overlapping capabilities. To draw a fair comparison between these two arbitrarily generated layouts, we did not optimize machine allocations in either case. Machines that share capabilities (having common REs) were placed

in close proximity in the functional layout, and were distributed arbitrarily in the distributed layouts. Recall that our intention in solving Problem 1, which poses four levels of overlapping capabilities (Table A.1) and six layouts (Table A.5), was to optimize material handling and other cost elements. Table 5.1 indicates that using distributed layouts results in significant savings. It is important to note the remarkably large cost reduction in case 1. These savings reflect the significant reduction in material handling costs that results when several machine tools with a number of shared capabilities are distributed, making their capabilities easily accessible from different regions of the layout. As we expected, the reduction in cost savings decreases as we move from case 1 to case 4. Our study thus shows that distributed layouts would be highly desirable in situations where there are many machine tools with several shared capabilities. Given that many modern manufacturing facilities contain a variety of machine tools with similar and overlapping capabilities able to produce a wide spectrum of components (Gindy *et al.*, 1996), distributed layouts are more relevant than ever.

Table 5.1: Comparison between Distributed and Functional layouts in Problem 1

Objective function values				
	Levels of sharing processing capabilities(REs)			
	Case 1	Case 2	Case 3	Case 4
DL1	322120	399395	830240	764405
DL2	286450	308695	764215	829820
DL3	389805	465485	746280	805985
DL4	276190	344705	796730	746250
DL5	288405	365225	591335	794100
Average	312549	376701	745760	788112
Functional	857625	882465	907890	904005
Percentage Saving	63.65	57.31	17.85	12.82

The cost savings under case 1 in Problems 2 to 6 appear in Figure 5.1. The first graph (graph-a) shows that the percentage of savings decreases as the number of parts increases when using distributed layouts. However, since larger problems incur higher production costs, the monetary value of the savings rapidly increases as problems grow in size (see graph-b), making distributed layouts very appealing.



(a) Percentage Saving (b) Saving in monetary units

Figure 5.1: Cost saving in moving from functional to distributed layout in Problems 1 to 6 under Case 1

5.1.2. Static versus Dynamic Distributed Layout

In this Section we compare static versus dynamically reconfigured distributed layouts in four different cases of Problem 1 (as described in the previous Section) and several other problems. We solved the problems by prohibiting dynamic reconfiguration. Hence, in a static distributed layout, machine allocation is optimized to provide a robust layout which remains unchanged for the entire planning horizon. Table 5.2 provides the values of the objective function in the four cases of Problem 1, and the percentage of savings obtained by changing from static to dynamic distributed layout. The Table shows that dynamic reconfiguration can lead to significant cost savings when the manufacturing system has more unique machines with less shared capabilities, as in case 4. Conversely, there is less need for system reconfiguration when a manufacturing facility has machine tools with several shared capabilities, as in case 1. As can be seen in Table 5.3, we found similar results in several other problems. Figure 5.2 shows that when using dynamic reconfiguration, the percentage of savings tends to decrease as problem size increases. However, the actual manufacturing cost in larger problems is very high, and even a small percentage in savings can imply a very significant

monetary value.

Table 5.2: Dynamic versus Static distributed layouts in Problem 1

	Total costs			
	Levels of sharing machines capabilities (REs)			
	Case 1	Case 2	Case 3	Case 4
SDL	254135	305605	412680	495315
DDL	239582	290827	313988	335991
Saving %	5.76	4.83	23.91	32.16

SDL = Static Distributed Layout; DDL = Dynamic Distributed Layout

Table 5.3: Dynamic versus Static distributed layouts in Problems 2 to 6

Problem No.	Case 1	Case 4
2	0.0	21.3
3	2.4	17.1
4	2.6	21.0
5	0.0	13.0
6	1.3	16.3

5.1.3. Other Model Features

In this Section, we illustrate the benefits of incorporating workload balancing, production planning and subcontracting in the proposed comprehensive model. The sample results in Table 5.4 show the distribution of a workload that requires the use of resource element-1 (RE-1), which is available on each of machines 1 to 6. In the first row in this Table, workload balancing ($\Upsilon = 0.99$) results in a workload that is evenly distributed among all the machines. In the second row, in contrast, the workload is unevenly distributed on the six machines when the workload factor Υ is set to zero. These results reflect the importance of incorporating a workload balancing constraint in the proposed model. As Table 5.5 shows, incorporating one or both of production planning and subcontracting

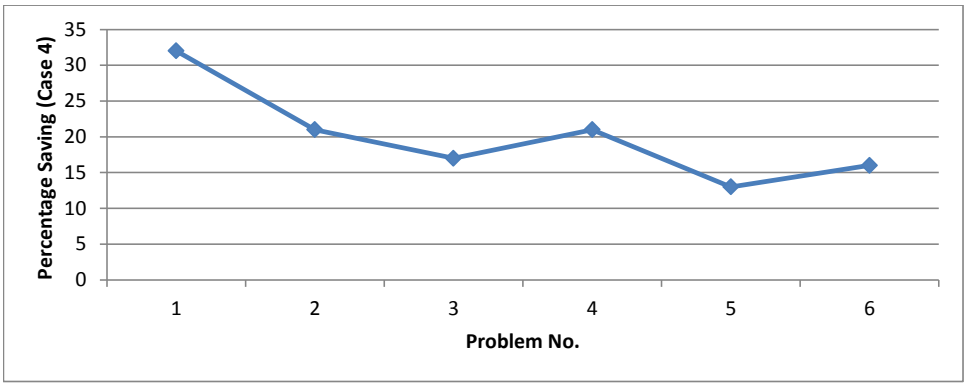


Figure 5.2: Cost saving percentage from dynamic reconfiguration as the problem size increases

typically result in a substantial decrease in the objective function, indicating their significance in economic terms. More importantly, the incorporation of these attributes affects several objective function terms, further signifying the value of utilizing a comprehensive model in manufacturing system analysis. A model consisting of different aspects of the system can help us to understand the problem better. An integrated system approach can minimize the possibility that certain important aspects of the system will be overlooked while other issues are being studied.

Table 5.4: Illustration of workload balancing

	Workload of RE-1 on Machines 1 to 6						
Υ	1	2	3	4	5	6	Total
0.99	1131	1131	1200	1131	1131	1131	6855
0.00	0	0	6350	0	750	0	7070

5.2. Performance Analysis

Due to the unique functionality of each type of metaheuristic, choosing an appropriate metaheuristic algorithm for solving a complicated mathematical model

Table 5.5: Effects of production planning and subcontracting

	Production Planning/Subcontracting			
Cost	without/without	with/without	without/with	with/with
Relocation cost	23220	30640	9100	10900
Material Handling cost	263105	207415	86555	58360
Inventory holding cost	0	23850	0	10150
Setup cost	19200	13050	11050	9900
In-house production cost	138500	138500	100400	103200
Subcontracting cost	0	0	157350	147450
Total cost	444025	413455	364455	339960

often is very demanding from a solution convergence, solution quality and algorithm robustness point of view. It is reflecting the user's experience and requires the examination of several algorithms and problems. In order to assess competing algorithms described in Chapter 4, we compare the LPSA with PSA in Section 5.2.1. Section 5.2.2 deals with the problem of comparisons between LPSA and LPGA algorithms and the results of LPSA are compared with MILP solver in Section 5.2.3. Section 5.2.4 is devoted to computing challenge we encountered in implementation phase and the final Section in this Chapter is concerned with dividing the search into phases.

5.2.1. Comparing LPSA with PSA

To gain some insight into the first set of our competing algorithms, Problem-1 was solved while considering the proposed mathematical model with and without workload balancing. By doing this, we can also examine the effectiveness of penalty approaches handling workload balancing and capacity constraints in PSA. Figure 5.3 and Figure 5.4 show convergence graphs for both LPSA and PSA algorithms solved Problem-1 with workload balancing factors $\Upsilon = 0$ and 0.80. As seen from Figure 5.3, PSA shows a comparable solution result to LPSA when workload balancing is not considered. However, PSA is not capable of solving the model considering a work load balancing constraint as illustrated in

Figure 5.4. Such poor performance of PSA search method could be in part own-

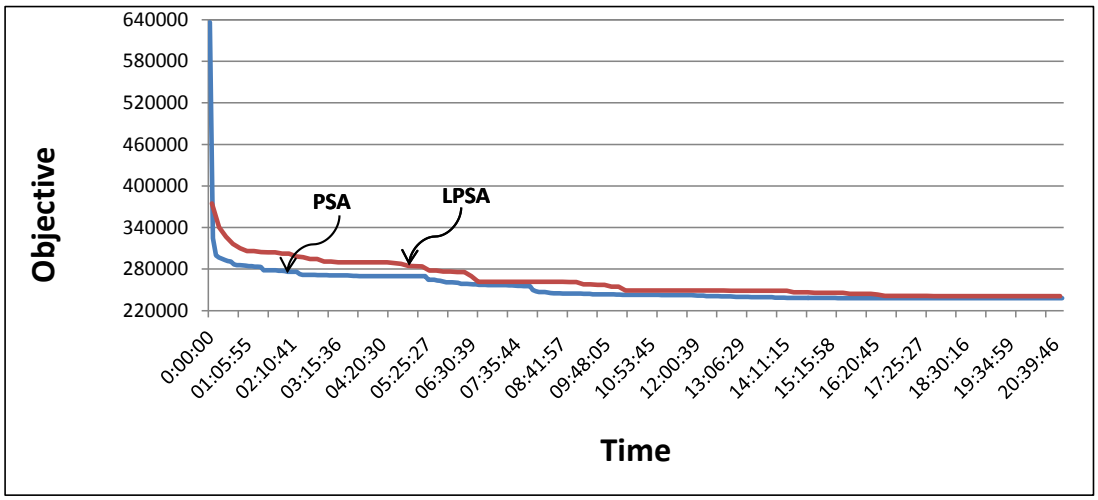


Figure 5.3: A convergence grahps of LPSA and PSA solved Problem-1 not considering a workload balancing constraint

ing to handling workload balancing using a penalty method which is an indirect way of constraint handling. Although a penalty function approach is generic and applicable to any type of constraint, its performance is not always satisfactory. The results reflects the fact that the penalty function methods often suffer the disadvantage of excessive fine tuning required by the penalty factors and lack of updating strategy for the penalty coefficient. On the other hand, the importance of balancing material flow to make the plant efficient has been advocated by several researchers. Greater resource sharing is believed to result in a more balanced workload among resources and improves overall utilization and reduces congestion. As the size of the problem increases, LPSA still outperforms PSA in terms of solution quality even when not considering a workload balancing constraint (see Figure 5.5). Another advantage of LPSA is its implementation simplicity compared with PSA employed a problem specific heuristic. In LPSA, designers easily define a valid combination of the values of the integer variables using

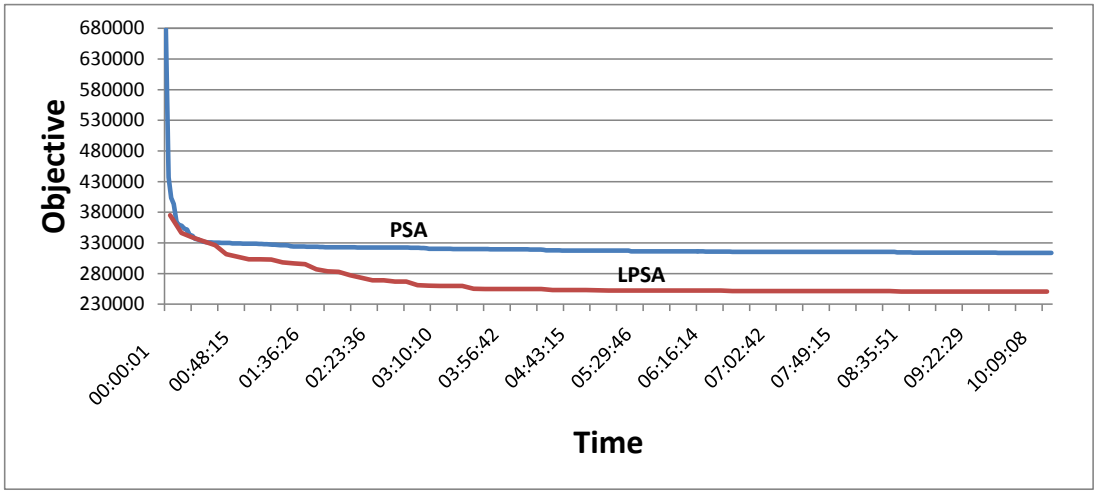


Figure 5.4: A convergence grahp of LPSA and PSA solved Problem-1 with a workload balancing constraint

solution representation and the values of the continuous variables which optimally correspond to a given combination of the integer variables are determined by solving a linear programming solver. It enables the algorithm being simple to implement in industrial setting, to re-implement by other researchers, and to explain and analyze. On the contrary, in the PSA, all constraints need to be addressed by structural properties of solution representation and some problem specific heuristics embedding in Simulated Annealing. For example, designers could not utilize a problem specific heuristic to address the production planning without good knowledge of lot sizing. Hence, the degree of complexity increases and makes PSA inferior to LPSA showing comparable results.

Moreover, despite the implementation simplicity, the LPSA presents less degree of complexity. According to Glover and Kochenberger (2003), a meaningful metric for algorithmic complexity is the number of parameters used in the algorithm. The reasoning behind that is the effort required to tune or understand these parameters is far greater as the number of parameters increases. In addition, larger parameters sets exhibit more complex parameter interactions and

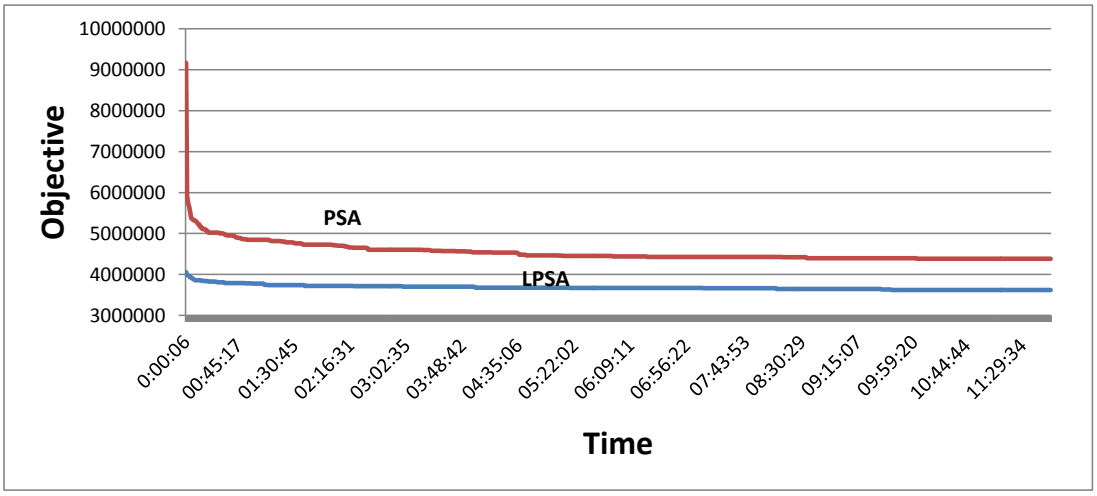


Figure 5.5: A convergence grahp of LPSA and PSA solved Problem-1 with a workload balancing constraint

might result in multiple locally optimal solutions in the parameter space. Tables 5.6 and 5.7 summarize all parameters used in LPSA and PSA. It can be seen that PSA uses a larger set of parameters and, thus, is more complex than LPSA.

5.2.2. Comparing LPSA with LPGA

Although the solution representation encoding integer variables of solution is identical in both LPSA and LPGA and they use a same approach to hybridize a metaheuristic with linear programming, their metaheuristic parts are incredibly diverse in nature. Such fact motivates us to make the comparison of their performance discussing in this Section. Figure 5.6-a illustrates convergence results for LPSA and LPGA algorithms when solving Problem-1. The result suggests that LPGA outperforms LPSA on small size problems in terms of solution quality. For fair comparison, the number of search directions in LPSA and population size in LPGA were assumed to be equal and set to 150. The examples also show that the developed LPGA is more efficient in solving the proposed model than LPSA in

Table 5.6: LPSA Algorithm Parameters

Category	**Description**	**Name**
General	Number of search directions	S
	Interaction frequency	F
	Number of iterations to be performed in each iteration	Q
	Number of iterations performed in each search phase (Dynamic or Static)	C_{phase}
Annealing Schedule	Initial temperature	T_0
	Cooling schedule exponent	α
Stopping criteria	Maximum number of iterations	N
	Iterations without improvement	-
Perturbation	Sublot perturbation probability	-
	Machine assignment perturbation probability	-
	Machine location perturbation probability	-
	Location perturbation probability	-

large size problem (see Figure 5.6-b). These results reflect the fact that, recombination operators in GAs enable them to mix good characteristics from different solutions to find solutions with better quality while SA does not gain much of this. A previously mentioned, in Simulated Annealing and Genetic Algorithms,

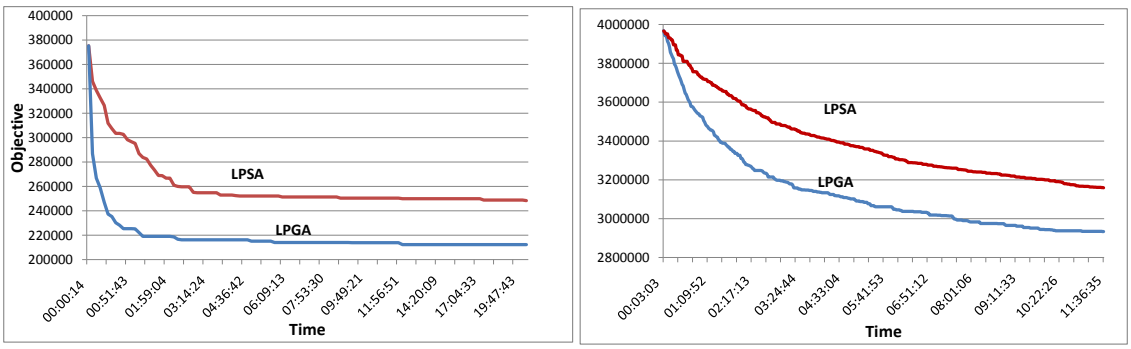


Figure 5.6: The convergence grahps of LPSA and LPGA solved small and large-sized problems

assigning different parameters values can lead to highly variable outcomes when different users run the same algorithms against the same problem. Thus, the algorithm robustness is an essential requirement for solution convergence. The

Table 5.7: PSA Algorithm Parameters

Category	**Description**	**Name**
General	Number of search directions	S
	Interaction frequency	F
	Number of iterations to be performed in each iteration	Q
	Number of iteration performed in each search phase (Dynamic or Static)	C_{phase}
Annealing Schedule	Initial temperature	T_0
	Cooling schedule exponent	α
Stopping criteria	Maximum number of iterations	N
	Iterations without improvement	-
Perturbation	Machine swap perturbation probability	-
	Alternative route perturbation probability	-
	Production setup perturbation probability	-
	Subcontracting ratio perturbation probability	-
	Sublot creation perturbation probability	-
	Sublot size perturbation probability	-
	Outsourcing perturbation probability	-

next numerical example is to investigate whether the convergence in the LPSA and LPGA are robust against different parameter settings for a given test problem. After computational testing on the different parameter settings for both algorithms, we selected the values shown in Tables 5.8 and 5.9 (which are robust across our test set). Figure 5.7 depicts a runtime comparison on results obtained using LPSA and LPGA algorithms to solve Problem-1. It can be seen that LPSA provides more degree of robustness compared with LPGA. This means that results in LPSA, in terms of accuracy, are more relatively insensitive to choice of parameters and thus less reliance on parameter tuning.

Table 5.8: Parameter settings for 6 different test cases on Problem-1 in LPSA

Test NO.	Parameters related to:								
	General			Perturbation Operators				Cooling Schedule	
	S	F	Q	p_1	p_2	p_3	p_4	T_0	α
1	80	10	10	0.025	0.1	0.15	0.15	30000	0.999
2	100	10	10	0.1	0.025	0.05	0.1	5000	0.95
3	120	10	10	0.15	0.05	0.05	0.15	20000	0.95
4	150	15	15	0.05	0.15	0.1	0.1	30000	0.99
5	170	15	15	0.025	0.1	0.15	0.05	10000	0.885
6	200	10	10	0.1	0.025	0.05	0.1	5000	0.95

Table 5.9: Parameter settings for 6 different test cases on Problem-1 in LPGA

Test NO.	Parameters related to:							
	General		Mutation Operators				Crossover Oper.	
	PopSize	TourSize	p_{m1}	p_{m2}	p_{m3}	p_{m4}	p_{cr1}	p_{cr2}
1	80	8	0.003	0.003	0.01	0.05	0.6	0.6
2	100	10	0.005	0.025	0.001	0.003	0.6	0.8
3	120	12	0.003	0.01	0.001	0.05	0.7	0.7
4	150	15	0.005	0.05	0.01	0.03	0.8	0.8
5	170	17	0.001	0.01	0.05	0.005	0.7	0.9
6	200	20	0.0025	0.025	0.01	0.05	0.6	0.5

The average convergence performance of 6 runs is graphed in Figure 5.7. This Figure indicates that LPGA were converged to better solution than LPSA after 840 generations.

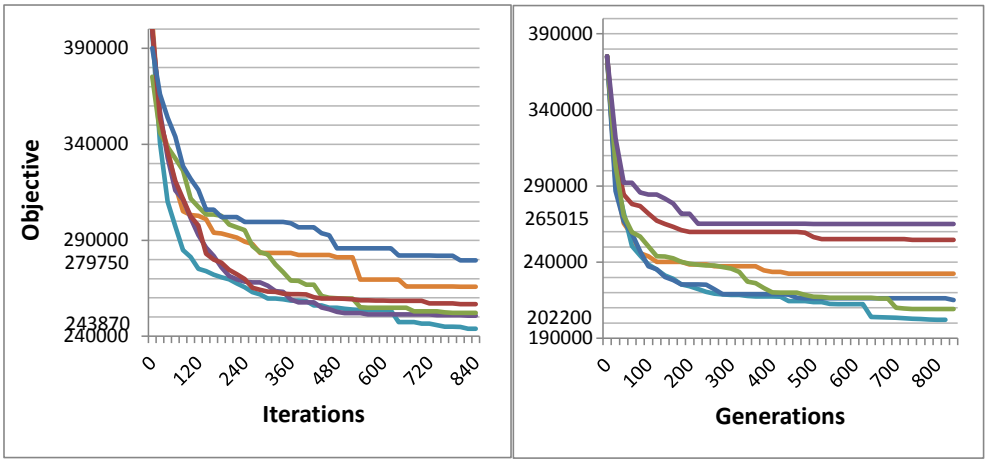


Figure 5.7: A comparison between convergence history of LPSA and LPGA solving small-sized problems using 6 different parameter settings given in Tables 5.8 and 5.9.

5.2.3. Comparing LPSA with MILP Solver

As we proposed a new solution procedure for a new mathematical model, there are no benchmark instances in literature to compare the solution quality of the developed algorithm. Instead, we made a comparison of the developed algorithm with the state-of-the-art optimization package - IBM ILOG CPLEX (version 12.2). The MILP model (to be solved by CPLEX) and the developed algorithm were implemented in a C++ programming environment and run on a personal computer having i5CPU@2.4GHz and 8 GB RAM. Figure 5.9 (a) shows the convergence history of CPLEX in solving Problem-1. From this Figure, it can be seen that the best solution found using CPLEX after 25 hours of computation has an objective function value of 451,485.00. This computation was continued for more than 100 hours and the solution did not improve. The same problem was solved using the developed algorithm and the convergence history is shown in Figure 5.9 (b). This Figure shows that the developed algorithm starts to provide solutions

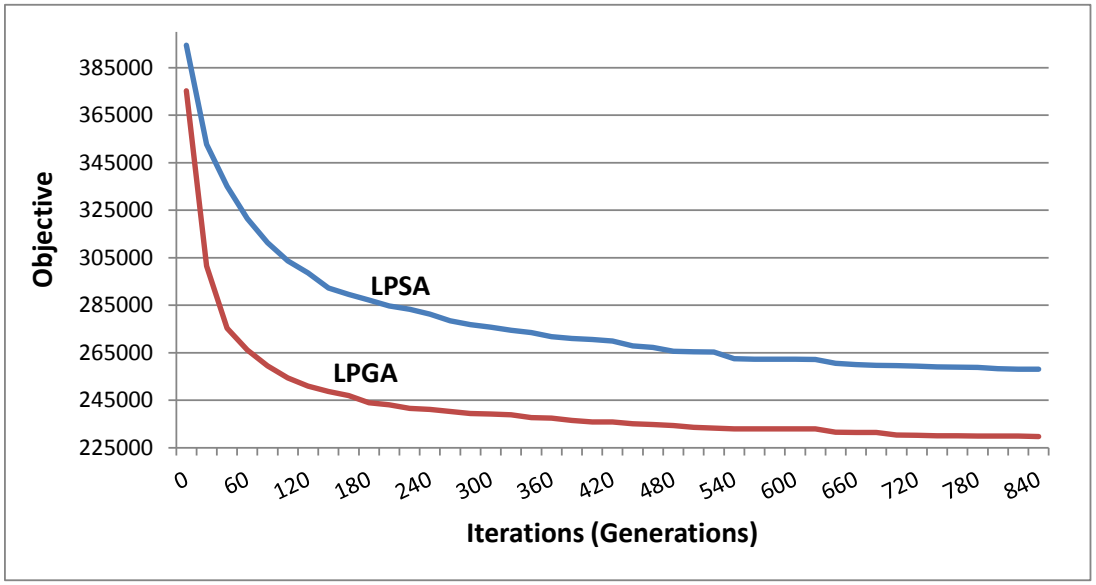
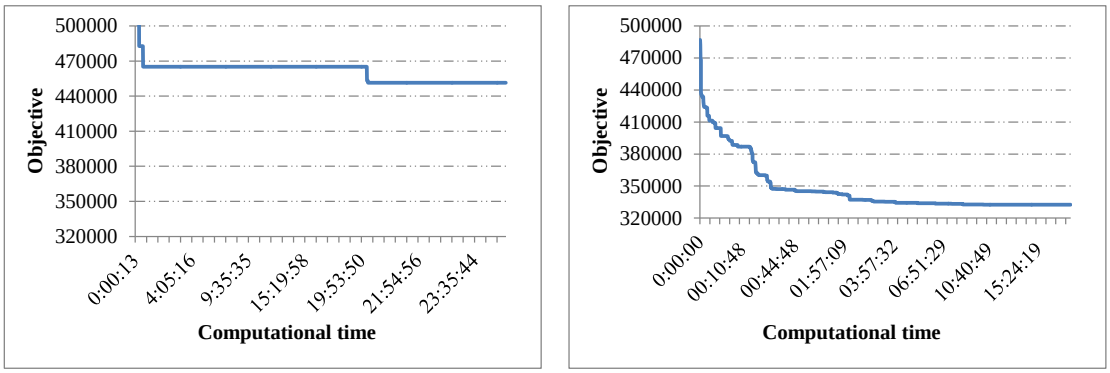


Figure 5.8: Average convergence performance of LPSA and LPGA given Figure in 5.7.

within a few minutes that are better than the one found after 100 hours computation using CPLEX. The final solution found using the developed algorithm has an objective function value of 332,485.00. In order to validate this improved solution generated by the simulated annealing, it was submitted to CPLEX as a starting incumbent solution. CPLEX accepted this improved solution as a feasible starting solution to the MILP model though CPLEX was unable to further improve it. For larger problems, CPLEX was unable to start computation because of large memory requirement whereas the developed algorithm was able to generate solutions in a minute and progressively improve those solutions. This demonstrates the potential of the developed algorithm in solving large problem instances.

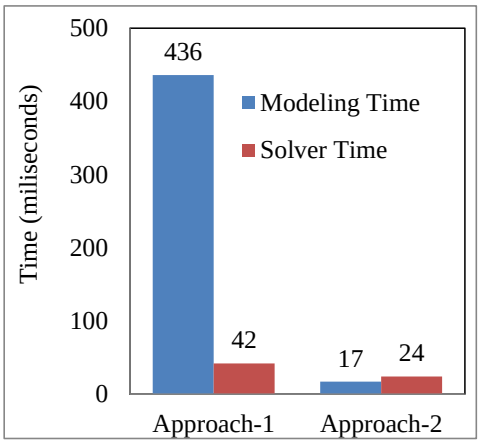


(a) CPLEX (b) Simulated Annealing

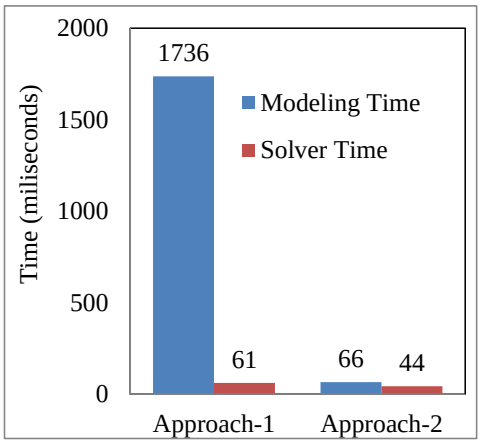
Figure 5.9: Computational performance of CPLEX vs the proposed LPSA

5.2.4. LP Implementation Approaches

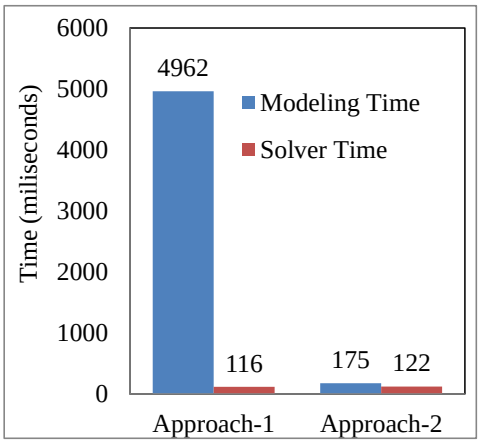
In Section 4.2.3, we discussed the computational challenge encountered in implementing the LP subproblem. In here, we exemplify this computational difficulty and the improvement obtained using the technique employed to circumvent this challenge. Figures 5.10 (a)-(d) show the computational times required in a single iteration along a single search path for four different problem sizes. The computational times are divided into modeling time (Step 3 in Figure 4.3) and solver time (Step 4 in Figure 4.3). These computational times are evaluated in two different approaches: Approach-1 (shown in Figure 4.4) and an improved approach - Approach-2 (shown in Figure 4.5). From Figure 5.10, it can be seen that the modeling time is many times larger than the solver time in Approach-1. This is a major computational hurdle. Approach-2 eliminates this computational difficulty by reducing the modeling time by about 96%. Without this improvement, the 20 hours convergence history shown in Figure 5.9 (b) would have taken more than 300 hours.



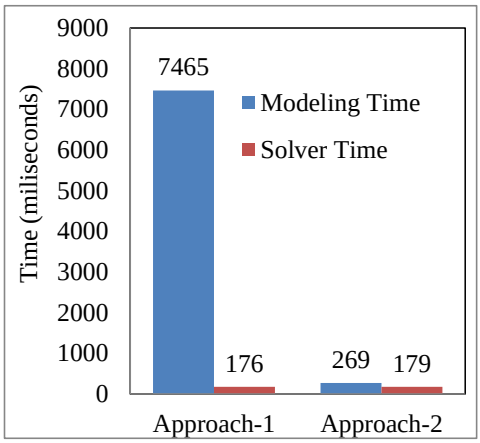
(a) $P = 25$, $\max\{O_p\} = 5$



(b) $P = 50$, $\max\{O_p\} = 6$



(a) $P = 80$, $\max\{O_p\} = 8$



(b) $P = 120$, $\max\{O_p\} = 8$

Figure 5.10: Average computational time for a single SA iteration in Approaches 1 and 2 for different problem sizes

5.2.5. Dividing the Search into Phases

The proposed LPSA algorithm first attempts to find solutions with static configurations and then it tries to improve these solutions by allowing dynamic system reconfiguration. The reasoning behind this approach was explained in Section 4.2.4. In here, we illustrate the advantage of this approach by examining the convergence of the algorithm in three different cases shown in Figure 5.11. Convergence curve (1) is when the algorithm started with initial solutions having dynamic reconfigurations and continued to improve this solution by running only

under phase-2. In convergence curve (2), the algorithm was started with initial solutions having static configurations and continued the iteration only under phase-1. Convergence curve (3) is when the algorithm started with initial solutions having static configuration, applied phase-1 up to the 5000^{th} iteration and then continued under phase-2. Looking into curve (1), it is evident that when the algorithm tries to find a solution with dynamic reconfiguration in one shot, it is unable to provide a solution better than the static solution found at the end of (2). Thus, without dividing the search into phases, a solution having the economic advantage of system reconfiguration can be unattainable. When the search is divided into phases as shown in curve (3), the algorithm is able to provide a solution possessing the economic advantage of dynamic reconfiguration.

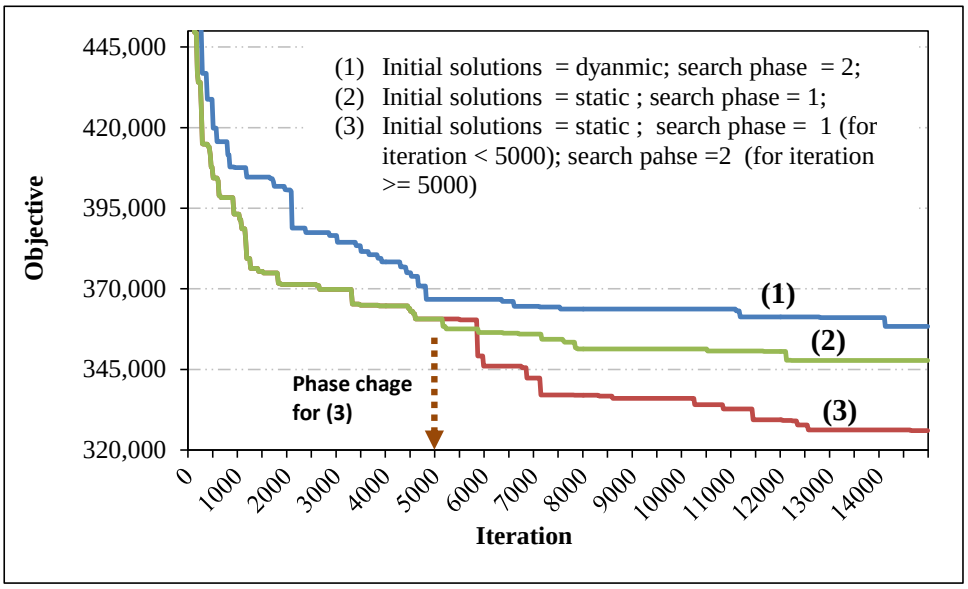


Figure 5.11: The impact of dividing the search into two phases on the convergence history and final solution quality

Part II

Scheduling of Manufacturing Systems with Distributed Layouts

Chapter 6

Literature Review: Flexible Job Shop Scheduling

In today's increasingly competitive and demanding marketplace, a high-performance delivery is a challenge. To address this challenge, scheduling can play a major role in a manufacturing process. A good schedule helps to eliminate excessive inventory cost, minimize setup cost, production time and provides accurate delivery date quotes. As defined in the literature, the allocation of jobs to resources to perform a collection of operations over a period of time is scheduling. Depending on shop environment, process constraints and performance criteria, obtaining a best schedule can be easy or difficult. Although some work in the literature have focused on distributed layout design (Montreuil and Venkatadri, 1991; Askin *et al.*, 1999; Benjaafar and Sheikhzadeh, 2000; Baykasoglu, 2003; Lahmer and Benjaafar, 2005; Hamedi *et al.*, 2012; Nageshwaraniyer *et al.*, 2013; Shafigh *et al.*, 2015), the scheduling problem in this manufacturing environment has not been addressed in the literature. In order to achieve the full capacity of a distributed layout with routing flexibility, an effective and efficient scheduling problem need to be formulated and solved.

In fact, there are some similarities between a distributed layout scheduling

problem and a flexible job shop scheduling problem (FJSP). Figure 6.1 illustrates the schematic representation of a DL consisting of several types machines distributed throughout the factory floor. If a complete sequencing is fixed and known for each job before scheduling is performed and also routing flexibility during scheduling is considered, the problem resembles flexible job shop scheduling problem. In FJSP, it is typical simplification assumption that distances between machines are ignored. However, machines with similar functionality are not necessarily located close in distributed layouts; thus, in addition to a time-based criterion (as main objective function in FJSP), traveling distance between machines also must be taken into account in scheduling of DLs. Sections 6.1 to 6.6 are devoted to brief review of job shop and flexible job shop scheduling modeling.

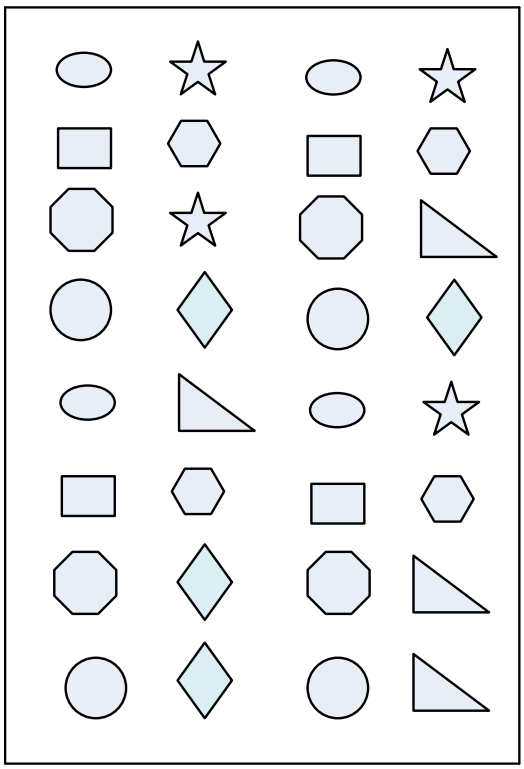


Figure 6.1: Shematic repressnetation of distributed layouts concept

6.1. Job Shop Scheduling Problem

The classical job shop scheduling problem (JSP) is one of the widely known and hardest combinatorial optimization problems, when a set of jobs must be processed on set of machines, where each job consists of number of operations. Each machine is capable to process one operation and the processing sequence of operations of each job is predefined. JSP aims to find the optimum allocation of the jobs to machines and starting times of jobs on each machine such that the performance indicator is optimized. In order to model JSP, the following assumptions are usually made:

1. Each machine can only perform one operation.
2. Preemption (cancelation of job at a time) is not allowed.
3. Jobs are independent and no priorities are assigned.
4. No blocking occurs due to finite buffer space.
5. The demand is known a prior and jobs are simultaneously available at time zero.
6. Precedence constraints among the operations of a same job are respected.
7. Each job can be performed by only one machine at time.
8. Breakdown is not considered.

Although mathematical formulation of machine scheduling problem is not promising solution method due to NP-complexity of the problem, it can shed light on developing an efficient heuristic and can be helpful to analysis the structure of the problem (Demir and İşleyen, 2013). Initial formulations of JSP were developed in 1960s. Using integer programming, Wagner (1959); Bowman (1959); Manne (1960) presented three distinct methods of modeling sequencing problem in JSP.

These three approaches are differentiated by the binary variable types defined to capture sequencing decision. These binary variables include sequence-position, precedence and time-indexed. Because only sequence-dependent variable type's formulation is relevant to our work in distributed layout scheduling, the reader may supplement this thesis with other qualified source in JSP.
This type of model was first introduced by Wagner (1959) and is based on the concept that each machine has a fixed number of production runs R_m (where r = 1,2,.....,R_m) and each of these production runs can be assigned at most to one operation of each job; thus, the assignment of operations to production runs of a given machine determines the sequence of jobs on that machine (Bayat Movahed, 2014).

A mixed integer linear programming (MILP) model for JSP which formulated based on sequence-dependent binary variables type and notations used in this formulation are presented below. The model is adopted from Defersha and Chen's work (Defersha and Chen, 2010).

Parameters:

j	Independent jobs need to be scheduled in the system.
o	An ordered and fixed list of operations that form a job.
$T_{o,j,m}$	Time required by a machine m to perform an operation o of job j.
R_m	Maximum number of production runs of machine m where production runs are indexed by r or $u = 1, 2, ..., R_m$;
$P_{o,j,m}$	A binary data equal to 1 if operation o of job j can be processed on machine m, 0 otherwise;
Ω	Large positive number.

Variables:

Continuous Variables:

$c_{o,j,m}$ Completion time of operation o of job j on machine m;

$\widehat{c}_{r,m}$ Completion time of the r^{th} run of machine m;

c_{max} Makespan of the schedule

Binary Variables:

$x_{r,m,o,j}$ Binary variable which takes the value 1 if the r^{th} run on machine m is for operation o of job j, 0 otherwise;

$z_{r,m}$ A binary variable which equal to 1 if the r^{th} potential run of machine m has been assigned to an operation, 0 otherwise;

MILP Model

Minimize:

$$Objective = c_{max} \tag{6.1}$$

Subject to:

$$c_{max} \geq c_{o,j,m} \ ; \quad \forall(o,j,m) \tag{6.2}$$

$$\widehat{c}_{r,m} \geq c_{o,j,m} + \Omega \cdot x_{r,m,o,j} - \Omega \ ; \quad \forall(r,m,o,j) \tag{6.3}$$

$$\widehat{c}_{r,m} \leq c_{o,j,m} - \Omega \cdot x_{r,m,o,j} + \Omega \ ; \quad \forall(r,m,o,j) \tag{6.4}$$

$$\widehat{c}_{r,m} - B_j \cdot T_{o,j,m} \cdot x_{r,m,o,j} \geq \widehat{c}_{r-1,m} \ ;$$

$$\forall(r,m,o,j)|\{r < R_m\} \tag{6.5}$$

$$x_{r,m,o,j} \leq P_{o,j,m} \ ; \quad \forall(r,m,i) \tag{6.6}$$

$$\sum_{m=1}^{M}\sum_{r=1}^{R_m} x_{r,m,o,j} = 1 \ ; \quad \forall(o,j) \tag{6.7}$$

$$\sum_{j=1}^{J}\sum_{o=1}^{O_j} x_{r,m,o,j} = z_{r,m} \ ; \quad \forall(r,m,j) \tag{6.8}$$

$$z_{r+1,m} \leq z_{r,m} \ ; \quad \forall(r,m,i) \tag{6.9}$$

$$x_{r,m,o,j}, \text{ and } z_{r,m} \text{ are binary} \tag{6.10}$$

The constraint in Eq. (6.2) ensures that the makespan value of system must be greater than or equal the completion times of all the operations. Eqs. (6.3) and (6.4) dictate that if $x_{r,m,o,j}$ is equal to 1 the o^{th} operation of job j and r^{th} run of machine m (q,r,m) must start at the same time. Constraint in Eq. (6.5) enforces that the succeeding operation of any job cannot be started after the preceding operation is completed. Eq. (6.6) permits the processing of each operation on eligible machine. Eq. (6.7) ensures that each operation can be processed in at most one machine. Eq. (6.8) restricts that on each run of any machine one operation only can be assigned. $(r+1)^{th}$ order on a machine can be assigned to an operation if and only if machine order r is already assigned (Eq. (6.9)). Eq. (6.10) explains non-negativity conditions of the decision variables.

The above model set the scheduling goal to find a feasible schedule which minimizes the makespan or maximum completion time. This is amount of time required to complete all the jobs in the system. Researchers usually consider this objective whenever the emphasis is on machine utilization and production flow. Lateness, total weighted completion time and weighted number of tardy jobs are among other optimally criteria in JSP literature. Although makespan minimization may not be a promising theoretical objective function, it widely used in academic and industrial practice. It is mathematically simple to handle, and permits easy developing of mathematical model. Thus, it is one of the most broadly used criterions in scheduling research.

In modern production environment, flexible manufacturing systems (FMS) representing important new development in automated manufacturing have been adopted broadly. In addition, multi-purpose machine tool provides manufacturing firms with competitive capabilities. System that allows easy storage and retrieval of large equipment and machine tools has been introduced. With this emerging technology to support job routing flexibility, the greater emphasis on job routing flexibility has been placed in recent years. Job routing flexibility allows processing a part type using alternative machine in case of encountering unforeseen events like machine breakdowns, order cancelation and new arrival (Kesen and Güngör, 2012).

6.2. Flexible Job Shop Scheduling Problem

The flexible job shop scheduling problem (FJSP) is an extension of the JSP where routing flexibility during scheduling is considered. It makes FJSP a more complicated problem due to simultaneously consideration of the both job routing and operation scheduling. A Job routing problem deals with assigning each operation to each machine among a set of capable machines. The scheduling sub-problem addresses sequencing assigned operations on all machines. FJSP was first addressed in Brucker and Schlie (1990) where a polynomial algorithm for solving FJSP with two jobs is developed. Since then, many other models and solution procedures have been devised to furmulate and solve FJSP in the literature. The last two decades have seen a growth in the number of publications on the subject of FJSP. Some recent publications consist of Chen *et al.* (1999b), Kacem *et al.* (2002b), Xia and Wu (2005), Chen *et al.* (2007), Saidi and Fattahi (2007), Gao *et al.* (2007), Pezzella *et al.* (2008b) and Xing *et al.* (2008).

6.3. Comprehensive Model for FJSP

In order to deal with real problems, several researchers have developed models and solution procedures lead to more realistic FJSP. A comprehensive model comprised of a set of different aspects of the system can help researchers to understand the problem better. It can reduce the possibility of some vital parts of the system being ignored, while other issues are being studied (Chen, 2001). For example, scheduling decision can be influenced by due date requirements, machine release dates, job priorities, machine setup requirements, operation and material handling system. Thus, scheduling must be capable of capturing simultaneously these diverse attributes. There is clearly need to consider an integrated scheduling model to address multi-faceted nature of the real world. Some attributes used in recently published articles in FJSP include sequence-dependent setups on machine, attached or detached nature of setups, machine release dates, and time lags. The way of incorporating these attributes in FJSP which will discuss below, provides a means of understanding the detailed model for scheduling in distributed layouts manufacturing system presented in the next Chapter.

Sequence-dependent Setup

Sequence-dependent setup is important factor that frequently appears in various manufacturing environments and in machine scheduling problems. In this situation, setup operations depend on the immediate preceding operation on the same machine. Panwalkar *et al.* (1973) showed that a large portion of jobs requires sequence-dependent setups in job scheduling. Limitation on research on job shops scheduling with sequence-dependent setups due to the complexity of the problem is pointed in Defersha and Chen (2010). Unlike the classical model assuming that setup time is included in processing time, it must be explicitly included in model. The setup time can be sequence-dependent and denoted by

$S_{o,j,m,o',j'}$, where operation o' of job j' is the last operation processed on machine m. The following constraint can replace Eq. (6.6) to capture such constraint.

$$\widehat{c}_{r,m} - B_j \cdot T_{o,j,m} - S_{o,j,m,o',j'} - \Omega \cdot (x_{r,m,o,j} + x_{r-1,m,o',j'}) + 2\Omega \geq \widehat{c}_{r-1,m} \ ;$$
$$\forall (r, m, o, j, o', j') | \{(r > 1) \wedge ((o, j) \neq (o', j'))\} \tag{6.11}$$

In addition, inclusion of the below constraint to the model enforces that, if operation o of job j is assigned to a production run r of machine m, any succeeding operation o' of job j cannot be assigned to any earlier run r' of machine m.

$$x_{r',m,o',j} \leq 1 - x_{r,m,o,j} \ ; \quad \forall (r, r'm, o, o'j) | \{(o' > o) \wedge (r' < r)\} \tag{6.12}$$

We need to defined same constraint where any preceding operation o' of job j cannot be assigned to any later run r' of machine m.

$$x_{r',m,o',j} \leq 1 - x_{r,m,o,j} \ ; \quad \forall (r, r'm, o, o', j) | \{(o' < o) \wedge (r' > r)\} \tag{6.13}$$

Machine Release Date

It is a common assumption in scheduling research that each machine is continuously available at time zero. However, a common situation in industry is the desire to perform ongoing operations from preceding schedule since production environment are seldom found empty (Ruiz *et al.*, 2008). Machine release date shows when a machine is released from previous work and can start processing. Since the routing flexibility of jobs permits assignment of jobs to one of available eligible machines, the machine release date certainly affects the selection of an alternative machine. Because a better choice is to look for an alternative machine released sooner. If each machine is subject to a release date D_m when it will be available for processing jobs of the current schedule, a constraint in Eq. (6.14) can be added to the model. $S^*_{o,j,m}$ denotes the setup time of operation o of job j if it is the first operation to be processed on machine m.

$$\widehat{c}_{1,m} - B_j \cdot T_{o,j,m} - S^*_{o,j,m} - \Omega \cdot x_{1,m,o,j} + \Omega \geq D_m \ ; \quad \forall (m, o, j) \tag{6.14}$$

The constraint in Eq. (6.14) guarantees that the starting time of the setup for the first run (r=1) of machine m needs to be greater than the release date D_m of the machine.

Time Lag

Time lags prescribe that an operation of a particular job may not be started on machine until at least certain time has elapsed since completing the previous operation of the job. The attribute allows a realistic treatment of a variety of practical scheduling problems. For instance, rest periods are used in processing of pastry products at the different point in the production sequence. Another application is overlapping of production where starting of $o+1$ operation of job j on a machine is allowed if a certain minimum finished operation o of job j have been completed on another machine. The minimum backlog in such cases may be represented by a time lag of appropriate length (Johnson, 1959). This is particularly true when a particular machine can be blocked due to limited buffer size and also in situation that the batch size is very large and there is a need to transfer a portion of the batch to the next machine (Defersha and Chen, 2010). Let's $L_{o,j}$ denotes the time lag. In order to ensure that starting time of (o, j) is greater than completion time of $(o-1, j) + L_{o,j}$ the constraint in Eq. (6.15) can be applied if run r' of machine m' is assigned to operation $o-1$ of job j and the first run of machine m is assigned to operation o of this same job.

$$\widehat{c}_{1,m} - B_j \cdot T_{o,j,m} - S^*_{o,j,m} - \Omega \cdot (x_{1,m,o,j} + x_{r',m',o-1,j}) + 2\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$

$$\forall (m, r', m', o, j) | \{((1, m) \neq (r', m')) \wedge (o > 1)\} \tag{6.15}$$

The constraint in Eq. (6.17) can also be used to utilize the same concept for other operation ($o > 1$) with time lag assigning in $r' > 1$.

$$\widehat{c}_{r,m} - B_j \cdot T_{o,j,m} - S_{o,j,m,o',j'} - \Omega \cdot (x_{r-1,m,o',j'} + x_{r,m,o,j} + x_{r',m',o-1,j}) + 3\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \; ;$$

$$\forall (r, m, r', m', o, j, o', j') | \{(r > 1) \wedge (o > 1) \wedge (r, m) \neq (r', m') \wedge (o, j) \neq (o', j')\} \tag{6.16}$$

Attached or Detached Setup

The model can permit anticipatory setups (detached) where the machine setup can be started before the corresponding job become available on the machine. In this situation, the setup time can overlap with processing time of previous operation in the sequence if these consecutive operations are not assigned in the same machine. Figure 6.2 illustrates a small job shop problem with two jobs which each has two operations processing by two machines. There is an anticipatory setup time between the jobs on the second machine. As can be seen in anticipatory part, the setup time for (o_2, j_1) can overlap with the previous operation's processing time because there is enough idle time at machine 2 to do setup before arriving (o_2, j_1). Conversely, a non-anticipatory setup (attached) requires the next operation of a job in the sequence to be already present at the machine 2 in order to perform the setup. For example, imagine the process of adjusting flat parts need a smooth finish on a surface grinding machine. This adjustment might require the flat part to be already present in order to firmly fix it to the machine. If the flat part is not arrived at the machine, the setup cannot carry out. A scenario is depicted in Figure 6.2 where the case of non-anticipatory setup is proposed.

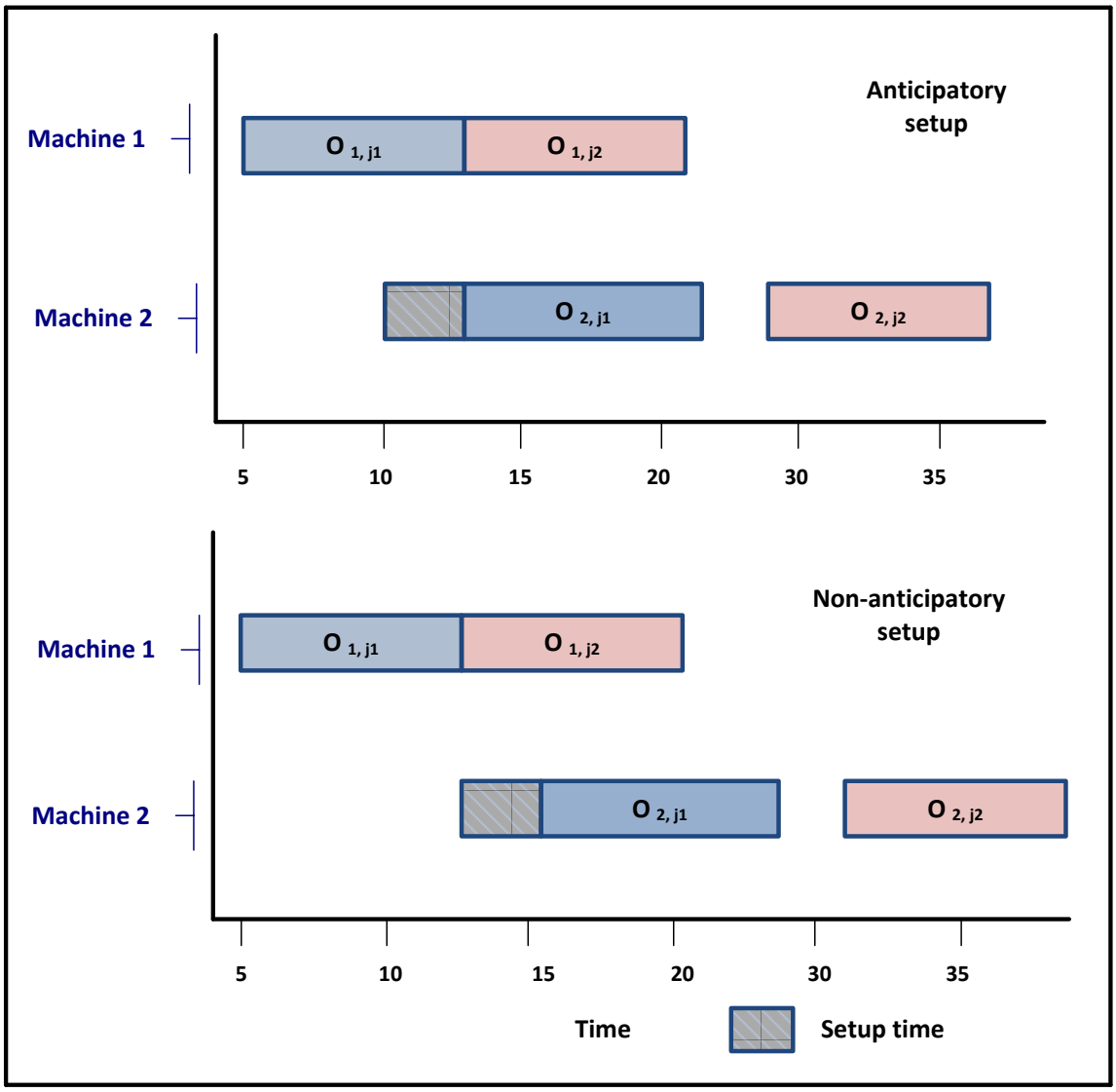


Figure 6.2: An example of anticipatory and non-anticipatory setup on a machine in job shop

Accordingly to Defersha and Chen (2010), the majority of the papers published in recent years proposed attached setup. However, Zhang and Gu (2008) claim that this assumption may hamper maximal concurrency, in turn, can significantly affects the research result in many situations. It is also important to point that in many productions environment a setup time can consume a significant portion

of the productive time if they are not handled with care (Framinan *et al.*, 2014). Let's introduce additional parameter $A_{o,j}$ as:

A binary data equal to 1 if the setup of operation o of job j is attached (non-anticipatory), or 0 if this setup is detached (anticipatory).

This parameter can be used to handle the type of setup by Eqs. (6.17) and (6.18)

$$\forall(r, m, o, j, o', j')|\{(r > 1) \wedge ((o, j) \neq (o', j'))\} \tag{6.17}$$

$$\widehat{c}_{1,m} - B_j \cdot T_{o,j,m} - S^*_{o,j,m} \cdot A_{o,j} - \Omega \cdot (x_{1,m,o,j} + x_{r',m',o-1,j}) + 2\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$

$$\forall(m, r', m', o, j)|\{((1, m) \neq (r', m')) \wedge (o > 1)\} \tag{6.18}$$

$$\widehat{c}_{r,m} - B_j \cdot T_{o,j,m} - S_{o,j,m,o',j'} \cdot A_{o,j} - \Omega \cdot (x_{r-1,m,o',j'} + x_{r,m,o,j} + x_{r',m',o-1,j}) + 3\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$

6.4. Objective Function in FSJP

Most of papers published in scheduling literature are based on minimization of single criterion or objective. However, similar to operational constraints, scheduling performance indicators might be varied and versatile in a real-life manufacturing scheduling scenario. Hence, in addition to satisfying operational goals (constraints), an objective function need to appropriately includes more criteria because managers routinely want to identify multiple criteria when assessing the schedule goodness. On the other hand, several researchers advocated that scheduling criteria may be conflicting, i.e. trying to increase machine utilization will results in high flow time. Many other examples can be given. Therefore, single criterion optimization problems usually fail to solve the problem, because it formalizes the best decision whose criterion value is not balanced with value of the constraints. To avoid such a failure, multi-criteria optimization techniques are used to deal with several objectives simultaneously. They provide an effective tool to formalize decisions considering different requirements and leading to a balance of conflicting goals. However, because of complexity and computational

difficulties, it is desirable to consider few objective terms in mathematical model. It is important to note that scheduling is related to short-term decisions, and it is rarely connected with long-term or medium-term strategy. Hence, rather than selecting long-term or medium -term goals, formalizing of such problems by specifying the short-term decision results in more efficient solution. One typical example would be the minimization of material handling distance. It is a typical assumption in FJSP that material handling distance is ignored because individual machines are usually close and working parallel in flexible job shop environment. This objective could be rarely more substantial than fulfilling due dates as a short-term objective. As result, in this case, there is no need to consider material handling distance and optimize both objectives simultaneously.

Multi-objective Optimization

Depending on how the different objectives are considered in the model, different class of multi-objective models can be formulated. The most employed models in manufacturing scheduling can be categorize in: (1) Weighted combination of objectives, (2) Lexicographical optimization, (3) Goal programming, (4) $\epsilon - constraint$ optimization, and (5) Pareto optimization classes. A review and discussion of last four classes is beyond the scope of this thesis, and we refer the reader to the literature for more comprehensive treatment.
The general multi-objective optimization problem is defined as follows:

$$\begin{aligned} \text{Minimize: } & F(x) = [F_1(x),\ F_2(x), ...,\ F_l(x)]^T \\ \text{Subject to: } & g_j(x) \leq 0;\quad j = 1,\ 2, ..., \ n \end{aligned} \tag{6.19}$$

where l is the number of objective functions and n is the number of inequality constraints. $x \in E^n$ is a vector of design variables, and $F(x) \in E^l$ is a vector of objective functions $F_i(x) : E^n \rightarrow E^l$. The feasible design space is defined as

$X = \{x \mid g_j(x) \leq 0, \quad j = 1, \ 2, ..., \ n\}$. The feasible criterion space is defined as $Z = \{F(x) \mid x \in X\}$.

In the weighted multi-criteria optimization, all objective functions can transfer to a single-objective using adding weight to each objective. The problem in Eq. (6.19) can be simplified to:

$$H = \sum_{i=1}^{l} w_i F_i(x) \tag{6.20}$$

If all of the weights are positive, as assumed in this study, then minimizing Eq. (6.20) gives a sufficient condition for Pareto optimality, which means the minimum of Eq. (6.20) is always Pareto optimal (Zadeh, 1963). Solution-X is Pareto optimal if no other solution has a better value than X for at least one objective and is not worse than X for the remaining objectives.

An example of design space for a multi-objective problem proposed in Marler and Arora (2010) is shown in Figure 6.3. The problem is consisted of two objective functions with a $(F1, \ F2)$ objective vector and $(x_1, \ x_2)$ is a vector design variable.

A familiar example of a sum weighted optimization in scheduling is minimization of weighted combination of the average job flowtime $\overline{F}$ and schedule makespam C_{max}, i.e. $\omega \times \overline{F} + (1 - \omega) \times C_{max}$, where $0 \leq \omega \leq 1$ (Sivrikaya-Şerifoğlu and Ulusoy, 1998). The most important feature of this approach is simplicity and easy to understand because all objectives terms can be transformed to a single-objective (a weighted sum). In order to guide the search, the objective preference or weight need to be provided by user prior an optimization run. As result, a single solution already considers the user's preference is obtained in each run. A biggest limitation of this approach is that a priori information must be precise and must reflect the preferences of the user accurately (Framinan *et al.*, 2014). When selecting weights for the objectives, one needs to avoid blind use of the method. It is crucial that weights can accurately represent the relative importance of objectives. If the weights are selected properly, the objective function can have a gradient that parallel to the gradient of the preference function. Marler (2009a)

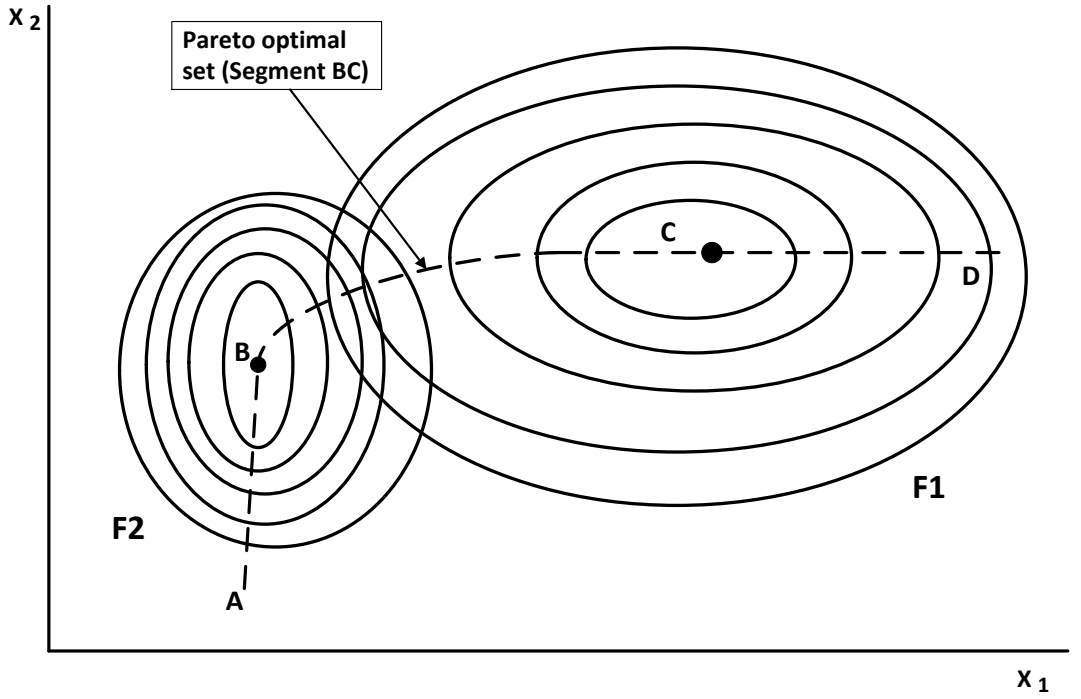


Figure 6.3: An example of design space in multi-objective optimization (Marler and Arora, 2010)

discussed the significant of selecting a set of weights because: (1) quantifying preference usually involves some degree of ambiguity, and (2) preference tends to be indistinct. Many researchers have developed systematic approaches to selecting weights. He provides a substantial look at these methods including rating, ranking, categorization methods and ratio questioning and paired comparison methods. In rating approach, as most common methods, one assign independent values of relative importance to each objective function. Some literature suggest that weight can be set such that $\sum_{i=1}^{k} \omega_i = 1$ and $\omega_i \geq 0$.

More importantly, when the objective functions have different range and magnitudes, one way is to transform the functions so that they do not naturally dominant (Marler and Arora, 2010). For example the objective functions can be

divided by their maxima to be normalized. By doing this, all weights are appropriately relative to each other and capable to reflect the relative importance of the objectives instead of the relative magnitudes of the function values.

Marler (2009b) presented and discussed various function transformation schemes in terms of potential numerical difficulties and in terms of imposed limit on function values. He evaluates these methods in terms of their capabilities to create an accurate representation of the Pareto optimal set using the weighted sum approach.

One of common approaches for function-transformation is the approach used in Koski (1981); Koski and Silvennoinen (1987) and Rao and Freiheit (1991) which is provided by Equation 6.21 where $F_i^{min} = minimum\ \{F_i(x) \mid x \in X\}$ and $F_i^{max} = maximum\ \{F_i(x) \mid x \in X\}$. F_i^{trans} generally has a value between zero and one, and denominator is guaranteed to be positive. This approach constrains the upper and lower limits of F_i^{trans} for each objective term i. Consequently it provides a relatively robust approach.

$$F_i^{trans} = \frac{F_i(x) - F_i^{min}}{F_i^{max} - F_i^{min}} \tag{6.21}$$

6.5. Material Handling Routes Consideration

In new generation manufacturing systems, shop floor control systems comprising methods used to prioritize, track and report against production orders and schedule are extensively utilized. It also involves procedures used to evaluate current resource status, labor, machine utilization, and other information required to support the overall planning, scheduling, and costing systems related to shop floor operations. The relationship between scheduling and shop floor control has been advocated by several researchers (Tuncel, 2012). Scheduling decisions that set goals for the shop are influenced by, and affect, the effectiveness of shop floor

control policies. If scheduling decisions are made poorly, the shop floor control systems may be inappropriately used. A large portion of the time a job spends on the shop floor is due to moving, this is clearly the case in distributed layouts manufacturing systems where machines with similar capabilities are distributed throughout shop floor. Thereby, an efficient a distributed layout scheduling model need to be capable of considering material handling distance as a vital performance measure for shop floor control function in distributed layout manufacturing systems. Thus, a key factor in linking scheduling and shop floor control decisions is the development of a accurate model to obtain a schedule that can also greatly enhance system performance. Kesen *et al.* (2010) demonestrated this concept in the context of virtual cellular manufacturing system where machines with similar processing abilities are distributed through the facility prior to forming virtual cells. The authors argue that because the same machine types are not necessarily located close to each other, traveling distance between machines must be considered. They developed a multi-objective MILP formulation considering two scheduling objectives, makespan and total traveling distance minimization. A sum weighted objective function which is the summation of weighted makespan and weighted total traveling distance values was considered in the model. In Mak *et al.* (2007), a mathematical model for virtual cell formation and scheduling problems was developed to minimize the total traveling distance incurred by parts. They conclude that an efficient manufacturing cells formation could reduce the cost of production schedule by minimizing material handling cost. Arkat *et al.* (2012) have presented an integrated model to concurrently solve the cells formation (the layout of machines inside cells), cellular layout (the layout of cells in the shop floor) and cellular scheduling problems with the objective of minimizing total transportation cost of parts as well as minimising makespan. In this article, the sum of intracellular and intercellular transportation costs was optimized.

6.6. Solution Procedure

It is well-known that the FJSP is NP-hard (Garey *et al.*, 1976). Like in FJSP, the distributed layouts scheduling problem is NP-hard due to the consideration of both routing of jobs and scheduling of operations. According to Byrne and Chutima (1997), when routing flexibility is embedded into the scheduling paradigm, the problem solution space is expanded owning to range of options created by the use of alternative routes. Therefore, the problem is very difficult to be solved by conventional optimization techniques. However, metaheuristic approaches are capable of producing reasonably schedules in relatively short time. Although metaheuristic usually do not offer solutions with a guaranteed distance to optimality, they can be powerful for most problem sizes. Parallelisation and cooperative computing are other solutions can be applied to complex scheduling settings which may be led to better results. In recent years, several metaheuristics such as tabu search, simulated annealing and genetic algorithms have been employed for FJSP. They can be categorized into two main classes: hierarchical approach and integrated approach. In the first approach, it takes advantages of the special structure of the FJSP by decomposing the problem into two subproblems: (i) machine assignment, and (ii) operation scheduling decisions to reduce difficulty. Once the assignment is done, the resulting sequence problem is JSP (Pezzella *et al.*, 2008a). This approach is adopted by Brandimarte (1993); Paulli (1995), among the others. The integrated approach can indeed give better results, as reported in (Vaessens *et al.*, 1996; Dauzère-Pérès and Paulli, 1997; Hurink *et al.*, 1994). In turn, it increases the computations involved.

As explained in Section 4.4.1, genetic algorithms were discovered as useful tool for a wide range of combinatorial optimization problems. Recently, GAs have been extensively used to solve the FJSP (Pezzella *et al.*, 2008a). Some important and relevant work are Chen *et al.* (1999a); Jia *et al.* (2003); Ho and Tay (2004); Kacem *et al.* (2002a); Pezzella *et al.* (2008a) and, in particular, Defersha

and Chen (2010) that gives the basis of our work. They have proposed all integrated approaches where solution representation, initial population generation, reproduction operators and chromosome selection procedure are only different. An appropriate coding scheme is a determinant of the GA behavior and help to find near-optimal solution for FJSP. Often, it cannot be easy to find a representation respects the structure of the search space and reproduction operators. The solution representation used in GA is typically tailored to the problem domain. The Section 6.8 reviews some important and relevant work proposed chromosome representations that have been used by GAs to solve the FJSP efficiently.

6.7. Transportation Constraints

In the most of machine scheduling models, it is assumed that transportation time is negligible because of existence of an infinite number of transporters for delivering jobs. Another reason for that is that jobs are assumed to move instantaneously from one machine to another machine. However, in many real life situations the time taken to transport jobs can significantly influence the completion time of the jobs. Considering transportation constraints in the scheduling formulation has been investigated by several researchers. Basically, two types of transportation constraints can be taken into account in developing a scheduling model. First, there is a transportation time such that jobs cannot instantaneously move from machine to machine. In this situation, transportation time could be modeled simply as a minimum time lag with the time it takes to carry out jobs between machines (Mitten, 1959; Langston, 1987). However, sometimes lot streaming is considered in the model to enable jobs to be split and transported from one machine to next one. It allows the overlapping performing of jobs on consecutive machines (Defersha and Chen, 2009b; Chan *et al.*, 2004; Dauzere-Peres and Lasserre, 1997). The second type of transportation constraint concerns to the

limitation of number of transporters. This means that if all transporters are busy in moving products, then the job has to wait until one becomes free. As result, transportation time is not fixed and it depends on both how the scheduling is being done and the traffic in the transportation (Framinan *et al.*, 2014). In contrast to conveyorized production system, assuming uninterrupted availability of the material handling equipment is not reasonable for systems which use AGV-based material handling (Sabuncuoglu and Hommertzheim, 1992). The most relevant works in the second type transportation constraint handling are those of Abdelmaguid *et al.* (2004); Bilge and Ulusoy (1995); Sabuncuoglu and Hommertzheim (1992).

6.8. Chromosome Representation

Several research results have shown that a better efficiency is obtained in GA search when the chromosome representation and its related operators are well designed to generate feasible solutions and avoid repair heuristic. The solution might be biased toward a certain region of the search space by using a repair mechanism, leading to the unbalanced distribution of the solution. Therefore, it is not always advisable to employ repair algorithm (Coello, 2002).

As previously mentioned, FJSP is a combination of assignment and scheduling decisions. The encoding idea of FJSP was first introduced in Cheng *et al.* (1996)'s research work on a tutorial survey of GA for JSP. Chromosome scheme in Chen *et al.* (1999a) is comprised of two integer strings with the total number of operations in length h. As can be seen in Figure 6.4, one string (String-1) encodes the assignment of a machine index to each operation such that the value of the j^{th} position of the string indicates the machine processing the j^{th} operation. Another string is given in Figure 6.4 as String-2 where the sequence of operations on each machine is encoded. O_{Mm} is an ordered set of operations on machine

M_m.

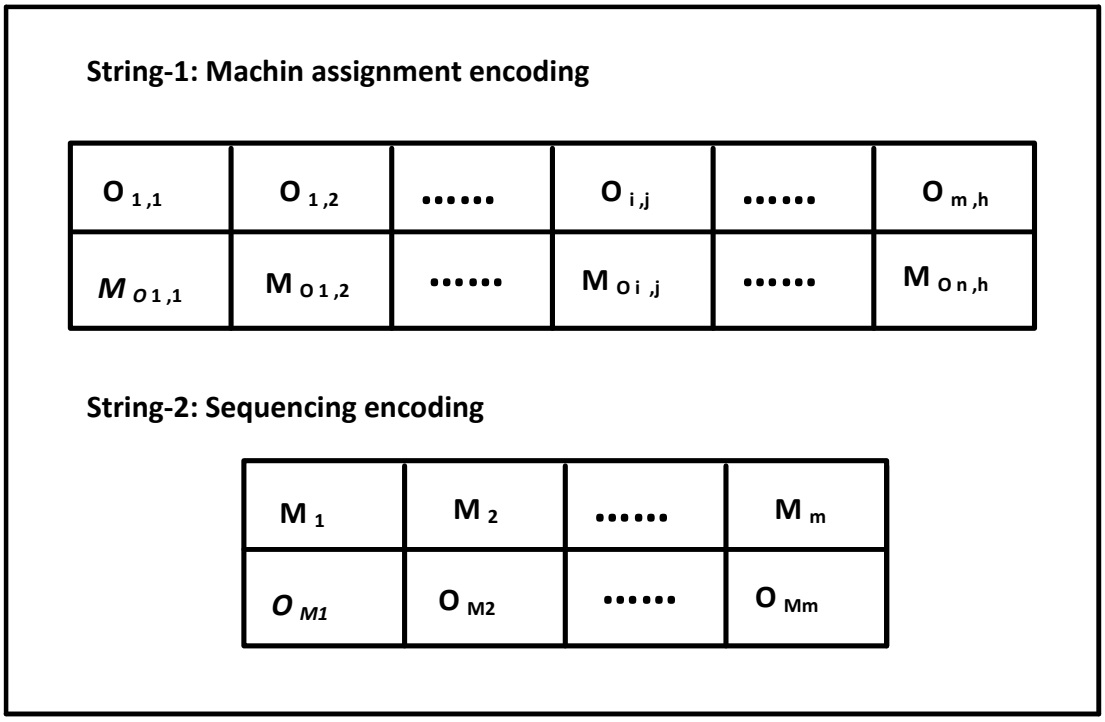


Figure 6.4: Chen *et al.* (1999a)'s encoding scheme for a FJSP.

However, this encoding generates invalid schedules and requires a repair mechanism to consider the sequence of operation. Paredis (1992) also splits solution representation into two parts, the first defines the routing policy and is identical to machine assignment string in Chen *et al.* (1999a), and the second encodes the order of operations on each machine. The String-2 in Figure 6.5 gives the sequence of any pair of operation in a set of values such that $a_{i,j,i,k}$ is equal to zero if the first operation $(O_{i,j})$ in the paired-combination is processed before the second operation$(O_{i,k})$.

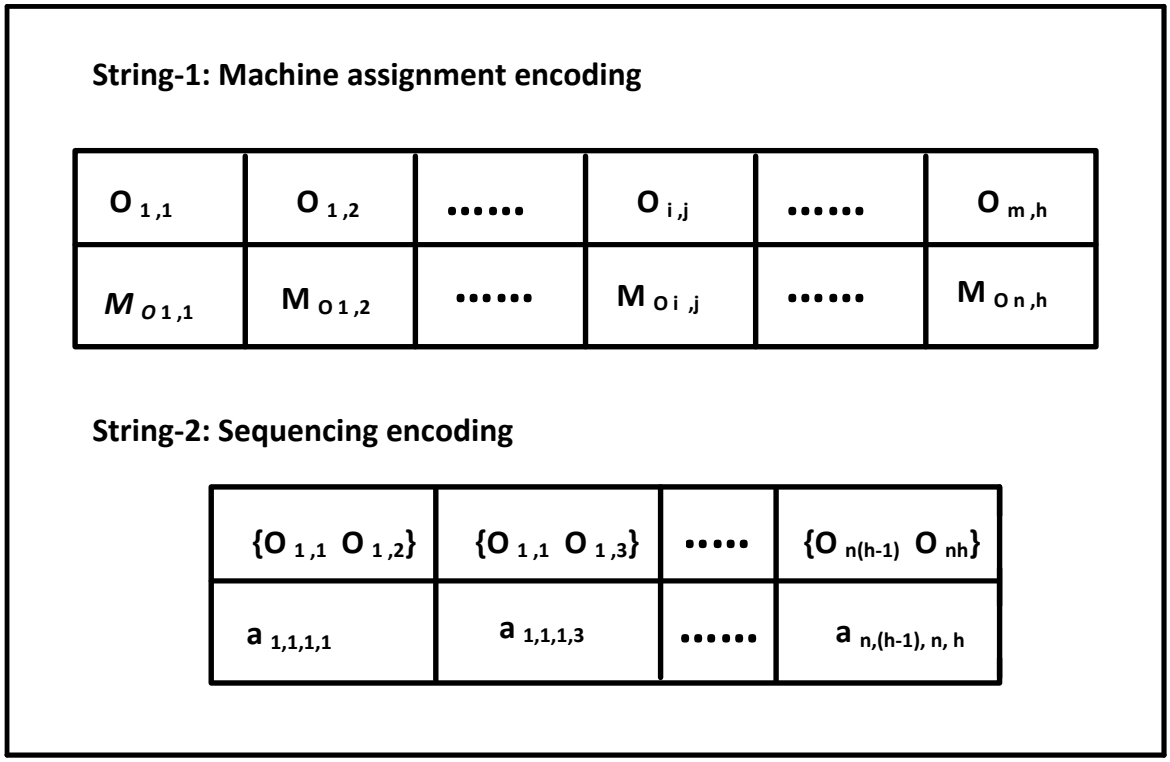


Figure 6.5: A two-string representation of a FJSP (Paredis, 1992)

Mesghouni *et al.* (1997) uses a parallel job representation using a matrix where each row is an ordered series of the operating sequence of a part and each element of this row including two terms. The first one is the index of machine performing this operation and second one is the starting time of this operation if the assignment of this machine on this operation is definitive. One example of possible chromosome representation is given in Figure 6.6.

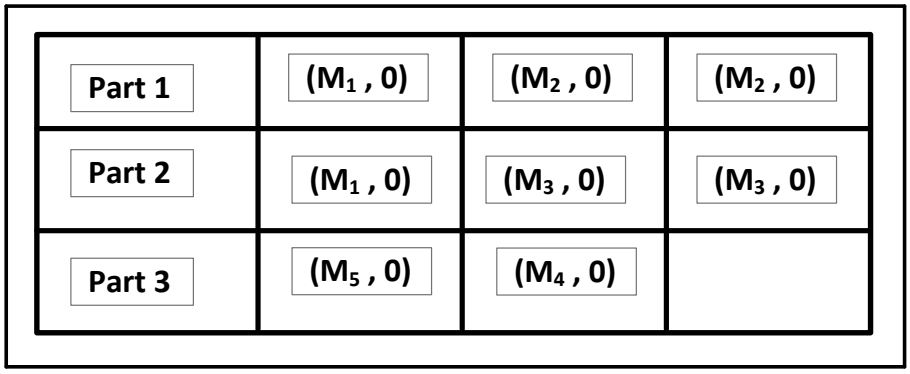


Figure 6.6: A parallel jobs representation scheme (Mesghouni *et al.*, 1997).

When a crossover operator applies on parents, the starting time of each operation on each machine could be invalid, thus, the children need a repair mechanism to recalculate the starting time for all operation. Consequently, the decoding complexity leads to computational difficulty in order to obtain even near-optimal solutions.

Kacem *et al.* (2002a) uses an assignment table representation that combines both routing and sequencing information. The table representes the schedule in table $S = (S_{i,k,j})$. If the $S_{i,k,j} = 0$ it indicates that the k^{th} operation of job i is not performed on machine j. In case M_m (machine index) is assigned for operation i, j, then $S_{i,k,j}$ is filled with pair $(s_{i,k}, c_{i,k})$ where $s_{i,k}$ is the starting time and $c_{i,k}$ is the completion time. One example of chromosome representation for small size problem is given in Figure 6.7.

		M_1	M_2	M_3	M_4
J_1	$O_{1,1}$	0	0	3 , 7	0
	$O_{2,1}$	4 , 7	0	0	0
	$O_{3,1}$	0	5 , 9	0	0
	$O_{4,1}$	0	0	0	10 , 15
J_2	$O_{1,2}$	0	9 , 14	0	0
	$O_{2,2}$	1 , 4	0	0	0
	$O_{3,2}$	0	0	1 , 3	0
J_3	$O_{1,3}$	0	1 , 5	0	0
J_4	$O_{1,4}$	0	0	7 , 14	0
	$O_{2,4}$	7 , 10	0	0	0
	$O_{3,4}$	0	0	0	1 , 10
	$O_{4,4}$	0	0	14 , 16	0

Figure 6.7: An example of table representation scheme proposed in (Kacem *et al.*, 2002a)

Compared to those described previously, this representation is more efficient since applying crossover and mutation never generate invalid schedules. The main disadvantages of this representation is that each assignment table must necessarily encode all machine set, thus, there are redundant assignments in the table leading to computational complexity. Ho *et al.* (2007) decodes solutions by using two

strings which is given in Figure 6.8. In the left hand side segment, the operation order is described while the right hand side segment encodes machine assignment in an array of binary values. By reading the data from left to right and increasing operation index of each job, a feasible schedule is always obtained from right hand side segment. For example, the operation sequence $2-2-1-1-2$ could be translated into a list of ordered operations: $O_{2,1}-O_{2,2}-O_{1,1}-O_{1,2}-O_{2,3}$ where $O_{i,j}$ denotes i^{th} operation in job j. In the left hand side, a machine among eligible machines which can perform an operation must be selected to get unit value. This solution representation is clear and direct, however, the binary coding resulting in increased consumption of memory space and computational time, especially in large size problem (Liu *et al.*, 2007).

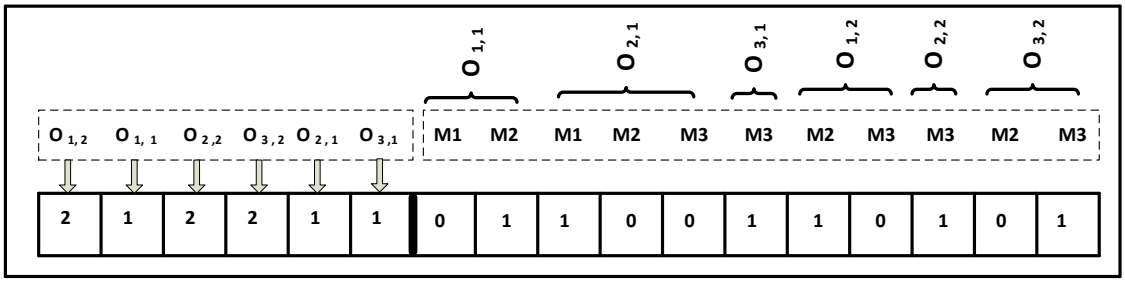


Figure 6.8: An example of encoding scheme proposed in (Ho *et al.*, 2007).

The chromosome representation in Pezzella *et al.* (2008a) is composed of several genes. Each gene encodes: (i) the assignment of operations to the machines, and (ii) the sequence of operations by the order in which they appear in the chromosome. Figure 6.9, for example, illustrates the chromosomal encoding of the initial solution for the small example given in Table 6.1. It is clear that the sequencing is feasible if it satisfies the precedence constraints among operations of the same job. The length of the string is equal to the total number of operations.

Table 6.1: An small example of assignment and sequencing solution

Machine	Sequence of Operations			
	Run1	Run2	Run3	Run4
M1	J3,O2	J4,O2		
M2	J1,O1	J1,O2	J2,O2	
M3	J2,O1	J3,O2		J1,O4
M4	J4,O1		J1,O3	

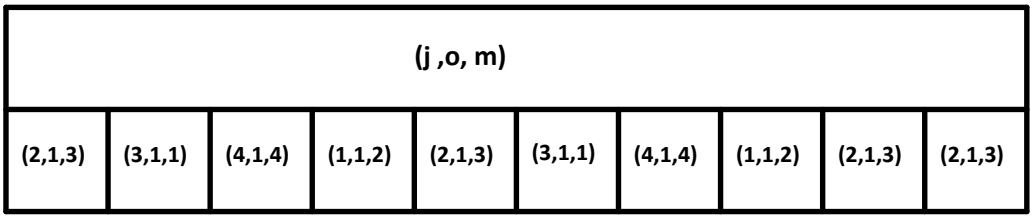


Figure 6.9: An example of encoding scheme proposed in (Pezzella *et al.*, 2008a).

Zhang *et al.* (2011) improves the solution representation in (Ho *et al.*, 2007) by employing an array of integer value to represent machine selection. As can be seen from the right hand side segment of chromosome representation in Figure 6.10, the length of array is the sum of all operations of all jobs. Instead of using binary value, M_m takes index of one of eligible machine to process a particular operation. Another segment (operation sequence) is identical to that proposed in (Ho *et al.*, 2007)'s work. M_m takes the value in the array of alternative machine set which can process a particular operation. For instance, M_1 is selected to process operation $O_{1,2}$ since the value in the array of alternative machine set is 1.

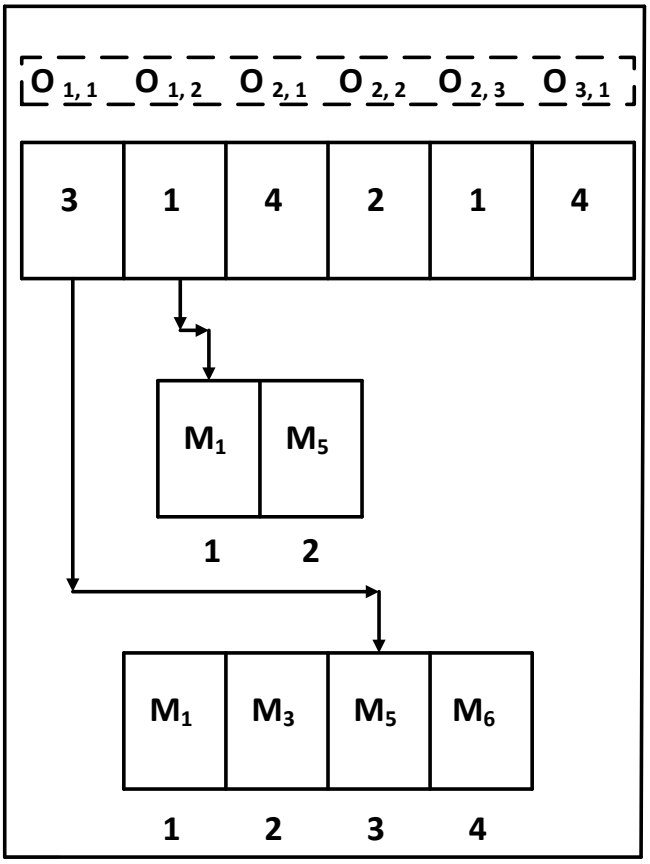


Figure 6.10: One possible encoding of the Machine Selection in Zhang *et al.* (2011).

Based on the analysis of the approach from the above literature, we adopt chromosome representation in Pezzella *et al.* (2008a) to reduce the cost of decoding. Because it benefits simple structure and encoding rule, and it requires no repair mechanism after recombination.

Chapter 7

Mathematical Model and Solution Procedure

7.1. Introduction

A mathematical model and solution procedure is described in this Chapter. The aim of the model is to determine an optimum schedule for a manufacturing system with distributed layout in a way that total transportation cost of parts and makespan are minimized. In order to efficiently solve the developed model, we also describe Multi Objective Genetic Algorithm (MOGA) in this Chapter.

7.2. Mathematical Model

The mathematical formulation for scheduling of manufacturing systems with distributed layouts is developed such that machine assignment and scheduling are simultaneously examined. The model is a multi-objective mixed integer programming model.

Assumptions

This model is developed under the following assumptions:

1. jobs are independent and no priorities are assigned.
2. Each job has no due date.
3. Each job can be performed by only one machine at time.
4. Recycling is not allowed in the model which means that each job can only visit a machine type at most once.
5. Jobs are produced in batch and batch sizes are known.
6. Transportation time is ignored.
7. Batch splitting is not allowed in the model.
8. Each job can be performed by only one machine at time.
9. The processing sequence of operations of each part is predefined and fixed.
10. The operating times for all operations on different machine are known.
11. Preemption (cancelation of job at a time) is not allowed.
12. All the machines that belong to a machine type are identical.
13. Machines with same factuality are located to different areas in the shop floor and they do not work parallel.
14. Each machine can only perform one operation at time.
15. The capabilities of each machine type are known and constant over time.
16. Breakdown are not considered.
17. No blocking occurs due to infinite buffer space.

Scheduling Objectives

As described in Section 6.4, multiple criteria should be considered in the objective function. However it is not possible to explicitly consider all criteria in the model due to the complexity and computational time required. In this research, objectives are limited to those which are also related the nature of manufacturing with distributed layout discussed in Chapter 6. Scheduling objective is weighted makespan of the schedule and total traveling distance minimization. One goal of the problem is to find a schedule of operations on machines (starting time of operation) which minimize the overall finishing time or makespan. Minimizing makespan yields to a good utilization of machines. Another one is to find assignment of jobs to the machines such that total distance traveled by parts is minimized. A weighted optimization technique is used to deal with these objectives simultaneously. Material handling cost is cost of transferring parts between machines. Unlike the flexible job shop environments, the same machine types are not closely located in the distrusted layouts. Because in this manufacturing environment machines that have the similar processing capabilities are located different areas in the shop floor to enhance the system's flexibility and efficiency. Therefore, traveling distance between machines must be considered in the scheduling to enhance system performance. The parts movements decrease the efficiency in distributed layout manufacturing system by increasing material handling requirements and flow time and complicating production control. These two scheduling criteria are interrelated and could be conflicting. The makespan can be minimized by assigning jobs to those machines are evenly distributed in the shop floor at the expenses of increased material handling cost.

Scheduling Decisions

The two decisions must be made during the design process: (1) the assignment of operations of each part type to machine types, and (2) the job start time at each machine and makespan or completion time of the last job to leave the system.

System and Input Parameters

The input parameter values must be supplied

1. **Jobs**: Independent jobs need to be scheduled in the system.
2. **Operation sequence**: An ordered and fixed list of operations that form a job.
3. **Operating time**: Time required by a machine to perform an operation on job.
4. **Machine type capability**: The ability of a machine type to perform operations.
5. **Machine distance**: Distances between each pair of machine.
6. **Transportation cost**: Unit transportation cost for each job to be carried between two machines.
7. **Available machines**: The available machines are the set of machines belong to different machine types.

Constraints

The following constraints must be imposed in the model.

1. A setup time which is sequence dependent so that for every machine there is a setup time that must precede the start of a given task that depends on both the job to be processed and the job that immediately precedes it. This time must be specified as input parameter.

2. Lag time must be specified at the beginning of the schedule. An operation of a particular job may not be started on machine until at least certain time (lag time) has elapsed since completing the previous operation of the job.

3. Each machine is not necessarily available at time zero. Machines release date must be specified to show when machines are released from previous work and can start processing.

Notation

Indices:

T Number of equal planning periods where planning periods are indexed by $t = 1, 2, ..., T$.

P Number of jobs where jobs are indexed by $j = 1, 2, ..., J$.

O_j Number of operations required by a job p where operations are indexed by $o = 1, 2, ..., O_j$.

M Number of machines in the manufacturing facility where machines are indexed by $m = 1, 2, ..., M$.

R Number of production runs of machine m where production runs are indexed by $r = 1, 2, ..., R$.

Input Parameters:

$T_{o,j,m}$ Unit processing time for operation o of job j on machine m.

$B_{o,j}$ Bach size of job j

$P_{o,j,m}$ A binary datum which equal to 1 if operation o of job j can be processed on machine m; 0 otherwise.

$A_{o,j}$ A binary datum which equal to 1 if the setup of operation o of job j is attached (non-anticipatory), or 0 if this setup is detached (anticipatory).

$E_{m,m'}$ Material handling distance between locations m and m'.

F_j Material handling cost per unit distance for one unit of job j.

$D_{m,}$ Release date of machine m when it will be available for processing jobs of the current schedule

$L_{o,j}$ Lag time for performing operation o of job j from the completion time of operation $o-1$.

$S_{o,j,m,o',j}$ Sequence-dependent set time for the setup of the machine m to perform operation o' if this operation is the last operation processed on machine m.

$S^*_{o,j,m}$ Setup time for the setup of the machine m to perform operation o if this operation is the first operation processed on machine m.

W_q weight of the q^{th} objective function j.

Ω Large positive number.

Decision Variables:

$c_{o,j,m}$ Completion time of operation o of job j on machine m;

$\widehat{c}_{r,m}$ Completion time of the r^{th} run of machine m;

c_M Makespan of the schedule

Binary Variables:

$x_{r,m,o,j}$ Binary variable which takes the value 1 if the r^{th} run on machine m is for operation o of job j, 0 otherwise;

$z_{r,m}$ A binary variable which equal to 1 if the r^{th} potential run of machine m has been assigned to an operation, 0 otherwise;

$d_{o,j}$ Distance between the machines where operations o and $o+1$ of job j are processed

Multi-objective Mixed Integer Linear Programming Model

Following the assumptions, scheduling objectives,constraints and using the notation given above, the proposed model is presented below.

Minimize:

$$Objective = \alpha \cdot c_M + (1-\alpha) \cdot \sum_{j=1}^{J} \sum_{o=1}^{O_j} (F_j \cdot d_{o,j}) \tag{7.1}$$

Subject to:

$$c_M \geq c_{o,j,m} \ ; \quad \forall (o,j,m) \tag{7.2}$$

$$\widehat{c}_{r,m} \geq c_{o,j,m} + \Omega \cdot x_{r,m,o,j} - \Omega \ ; \quad \forall (r,m,o,j) \tag{7.3}$$

$$\widehat{c}_{r,m} \leq c_{o,j,m} - \Omega \cdot x_{r,m,o,j} + \Omega \ ; \quad \forall (r,m,o,j) \tag{7.4}$$

$$\widehat{c}_{1,m} - B_j \cdot T_{o,j,m} - S^*_{o,j,m} - \Omega \cdot x_{1,m,o,j} + \Omega \geq D_m \ ; \quad \forall (m,o,j) \tag{7.5}$$

$$\widehat{c}_{r,m} - B_j \cdot T_{o,j,m} - S_{o,j,m,o',j'} - \Omega \cdot (x_{r,m,o,j} + x_{r-1,m,o',j'}) + 2\Omega \geq \widehat{c}_{r-1,m} \ ;$$

$$\forall (r, m, o, j, o', j') | \{(r > 1) \wedge ((o, j) \neq (o', j'))\} \tag{7.6}$$

$$\widehat{c}_{1,m} - B_j \cdot T_{o,j,m} - S^*_{o,j,m} \cdot A_{o,j} - \Omega \cdot (x_{1,m,o,j} + x_{r',m',o-1,j}) + 2\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$

$$\forall (m, r', m', o, j) | \{((1, m) \neq (r', m')) \wedge (o > 1)\} \tag{7.7}$$

$$\widehat{c}_{r,m} - B_j \cdot T_{o,j,m} - S_{o,j,m,o',j'} \cdot A_{o,j} - \Omega \cdot (x_{r-1,m,o',j'} + x_{r,m,o,j} + x_{r',m',o-1,j}) + 3\Omega \geq \widehat{c}_{r',m'} + L_{o,j} \ ;$$

$$\forall (r, m, r', m', o, j, o', j') | \{(r > 1) \wedge (o > 1) \wedge (r, m) \neq (r', m') \wedge (o, j) \neq (o', j')\} \tag{7.8}$$

$$x_{r,m,o,j} \leq P_{o,j,m} \ ; \quad \forall (r, m, i) \tag{7.9}$$

$$\sum_{m=1}^{M} \sum_{r=1}^{R_m} x_{r,m,o,j} = 1 \ ; \quad \forall (o, j) \tag{7.10}$$

$$\sum_{j=1}^{J} \sum_{o=1}^{O_j} x_{r,m,o,j} = z_{r,m} \ ; \quad \forall (r, m, i) \tag{7.11}$$

$$z_{r+1,m} \leq z_{r,m} \ ; \quad \forall (r, m, j) \tag{7.12}$$

$$x_{r',m,o',j} \leq 1 - x_{r,m,o,j} \ ; \quad \forall (r, r'm, o, o'j) | \{(o' > o) \wedge (r' < r)\} \tag{7.13}$$

$$x_{r',m,o',j} \leq 1 - x_{r,m,o,j} \ ; \quad \forall (r, r'm, o, o'j) | \{(o' < o) \wedge (r' > r)\} \tag{7.14}$$

$$x_{r,m,o,j}, \text{ and } z_{r,m} \text{ are binary} \tag{7.15}$$

The constraint in Eq. (7.2) ensures that makespan value of the system must be greater than or equal the completion times of all the operations. Constraints Eqs. (7.3) and (7.4) dictate that if $x_{r,m,o,j}$ is equal to 1 the o^{th} operation of job j and r^{th} run of machine m (q, r, m) must start at the same time. The constraint in Eq. (7.5) guarantees that the starting time of the setup for the first run (r=1) of machine m need to be greater than the release date D_m of the machine. Constraint Eq. (7.6) is to restrict that the setup of any production run $r > 1$ of a given machine cannot start before the completion time of run $r - 1$ of that machine. For any pair machine(m, m'), constraint in Eq. (7.7) equate that the setup or the actual processing of the first run on machine m cannot

start before the completion of run r' of machine m' plus the lag time $L_{o,j}$.This constraint is applied if run r' of machine m' is assigned to operation $o-1$ of job j and the first run of machine m is assigned to operation o of this same job. Constraint in Eq. (7.8) is similar to Eq. (7.7), except that Eq. (7.8) is for run $r > 1$ of machine m where setup time depends on the operation assigned to run $r-1$. Eq. (7.9) permits the processing of each operation on eligible machine.Eq. (7.10) ensures that each operation can be processed in at most one run of eligible machine.Eq. (7.11) restrict that on each run of any machine one operation only can be assigned. $(r+1)^{th}$ order on a machine can be assigned to an operation if and only if machine order r is already assigned(Eq. (7.12). Constraint Eq. (7.13) enforces that if operation o of job j is assigned to a production run r of machine m, any upcoming operation o' of job j cannot be assigned to any earlier run r' of machine m. Constraint in Eq. (7.14) is symmetric to constraint Eq. (7.13).

7.3. Multi Objective Genetic Algorithm

As noted in Section 6.6, it is proven that most of the scheduling problems are NP-hard. The proposed scheduling problem applied in manufacturing with distributed layout is even more complex due to the simultaneous difficulties of NP-hard complexity and of multi-objective framework. Therefore, heuristic algorithm must be used for large problems. In this study, a Multi Objective Genetic Algorithm (MOGA) is proposed to obtain a near optimal schedule in a manufacturing system with distributed layout. The objective criteria considered are minimizing makespan and minimizing total material handling cost. A single-objective genetic algorithm is extended to solve multi-objective optimization problem by combining the objectives into a scalar objective function. The weights used for combining multiple objectives into a scalar function are randomly specified for each selection to enable multidirectional search. A sum weighted objective function which is the

summation of weighted makespan and weighted total traveling distance values is used to transfer these two objectives to a single- objective function. The following sections deal with detailed MOGA including chromosome representation, genetic operators and fitness function.

7.3.1. Chromosome Representation

As discussed in Section 6.8, a better efficiency is achieved in GA search if the chromosome and its related operators are well designed to generate feasible solutions and avoid repair heuristic. Because the problem is a combination of assignment and scheduling decisions and based on discussion in Section 6.8, we adopt the solution representation from the work of Pezzella *et al.* (2008a); Defersha and Chen (2010) in our study. The chromosome scheme comprises of a string with the total number of operations in length. Each gene is described by a triplet (j, o, m) where m can assume the index of an alternative machine on which an operation o of job j can be assigned. In order to always have a feasible schedule, o must be defined as the progressive number of operation within job j. The order of the jobs in the machines is represented by the sequence of the genes in the string. A pictorial correspondence of this solution representation is given in Figure 7.1 where the problem instance of Table 7.1 is considered.

Table 7.1: An small example of the operation-machine assignment.

	Runs		
Machines	Run1	Run2	Run3
M1	(j3, o1)	(j1, o2)	
M2	(j1, o2)	(j3, o2)	
M3	(j2, o2)	(j1, o3)	
M4	(j2, o1)	(j3, o3)	(j2, o3)

In this Figure, production runs of machine $m = 4$ are illustrated as example.

The proposed encoding scheme seems to represent some permutations of operations. However, such permutations do not always represent feasible schedules since the precedence relationships among operations must be kept to be a feasible schedule. In genetic algorithms, it is desirable to initially have a population that consists of feasible chromosomes before the application of genetic operators (Lee *et al.*, 1998). To construct such an initial population, the following procedure is developed which creates chromosomes preserving the imposed precedence relationships by using the information of the precedence relation matrix.

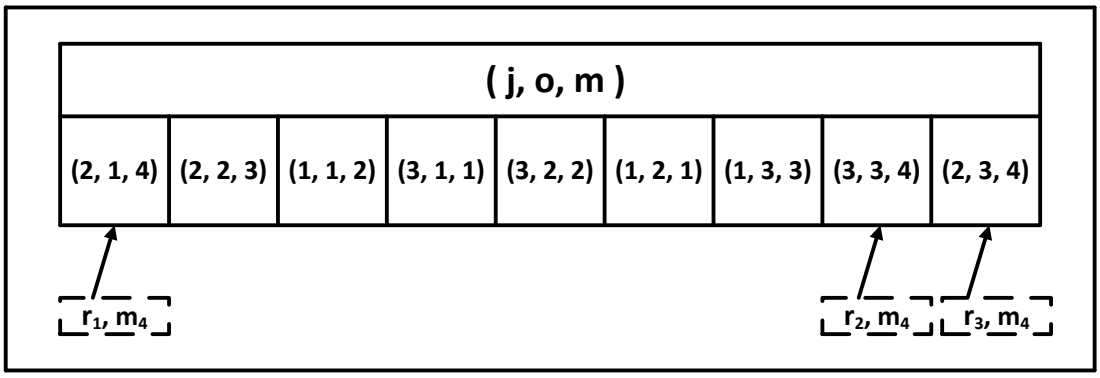


Figure 7.1: Choromosome encoding

7.3.2. Genetic Operators

In this study, we propose a k-way tournament selection scheme to select k chromosomes from population and choose one with highest fitness for reproduction. The process is repeated with replacement until a desired size of mating pool has been formed. This method is widely used in GA applications due to its efficiency and coding simplicity. Crossover operators are applied directly on the pairs of parent chromosomes identified from the selection step to give birth one or more offspring. The algorithm incorporates two problem specific crossover operators, assignment and sequencing operators. In the assignment operator, some machine

assignments alleles m from each parent are arbitrarily chosen and then they reproduce offspring by exchanging this assignment of a subset of operations between the two parents. Steps of applying the assignment crossover operator for each pair of parents are as follows:

Step 1. Randomly select some machine assignment allele m for each parent (As indicated by asterisks in Figure 7.2).

Step 2. Produce two offspring by copying all the genetic material of the parents to their respective child except the assignment property of the randomly selected operations.

Step 3. For completing the unassigned positions on the operation assignment of the first offspring, Place corresponding operation assignment of the second parent.

Step 4. For completing the unassigned positions on the operation assignment of the second offspring, place corresponding operation assignment of the first parent.

The procedure is illustrated in Figure 7.2. This operator is applied on each pair of parents with probability ρ_1.

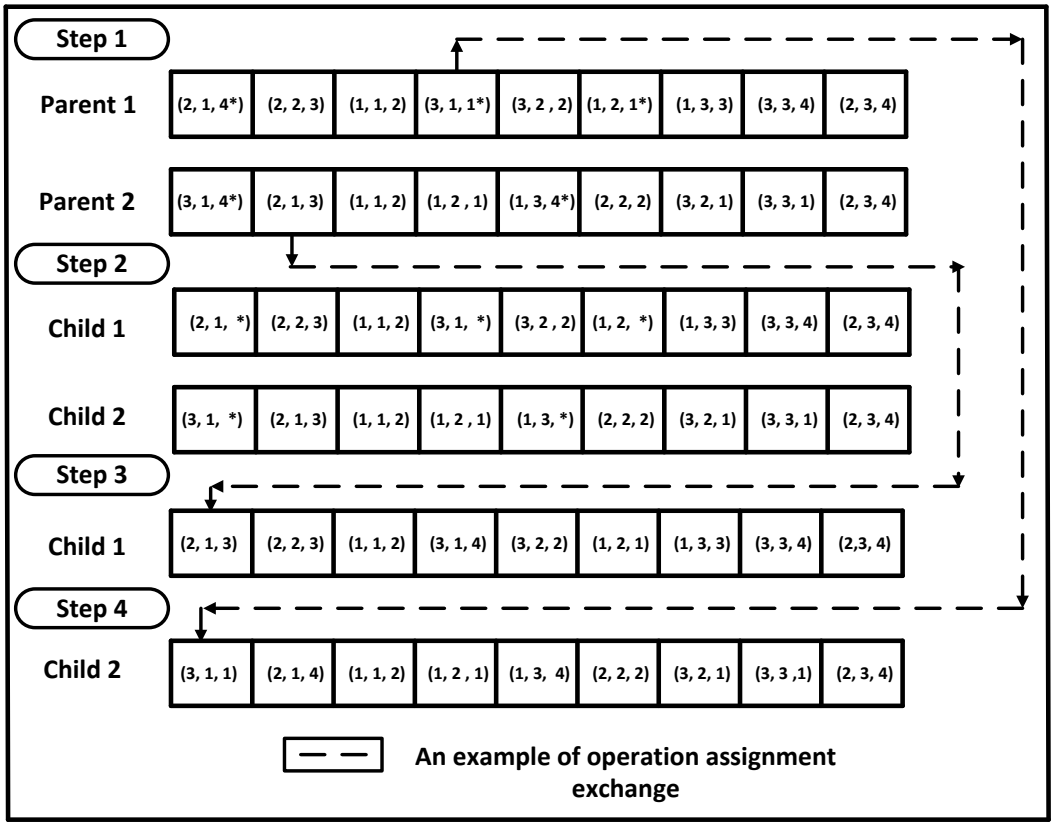


Figure 7.2: Assignment crossover

Sequencing crossover operator only changes the order of the operations in the parent chromosomes and the operation assignment is preserved in the offspring. The precedence preserving order-based crossover (POX) of Kacem *et al.* (2002a) is used in the proposed algorithm. This operator respects the precedence constraint among operations of the same job. As discussed in Section 6.8, a better efficiency is obtained in GA search when operators are well designed to generate feasible solutions and avoid repair heuristic. Hence, it is probable to design operators such that precedence constraints are not violated. Steps of the Sequencing crossover for each pair of parents are performed as follows:

Step 1. Randomly select one operation for first parent.

Step 2. Copy all operations of the job to which the selected operation belongs to its child.

Step 3. Complete the offspring with the remaining operations, in the same order as they appear in the second parent, while maintaining the assignment property of the operations in the first parent.

The procedure is shown in Figure 7.3. This operator is applied on each pair of parents with probability ρ_2.

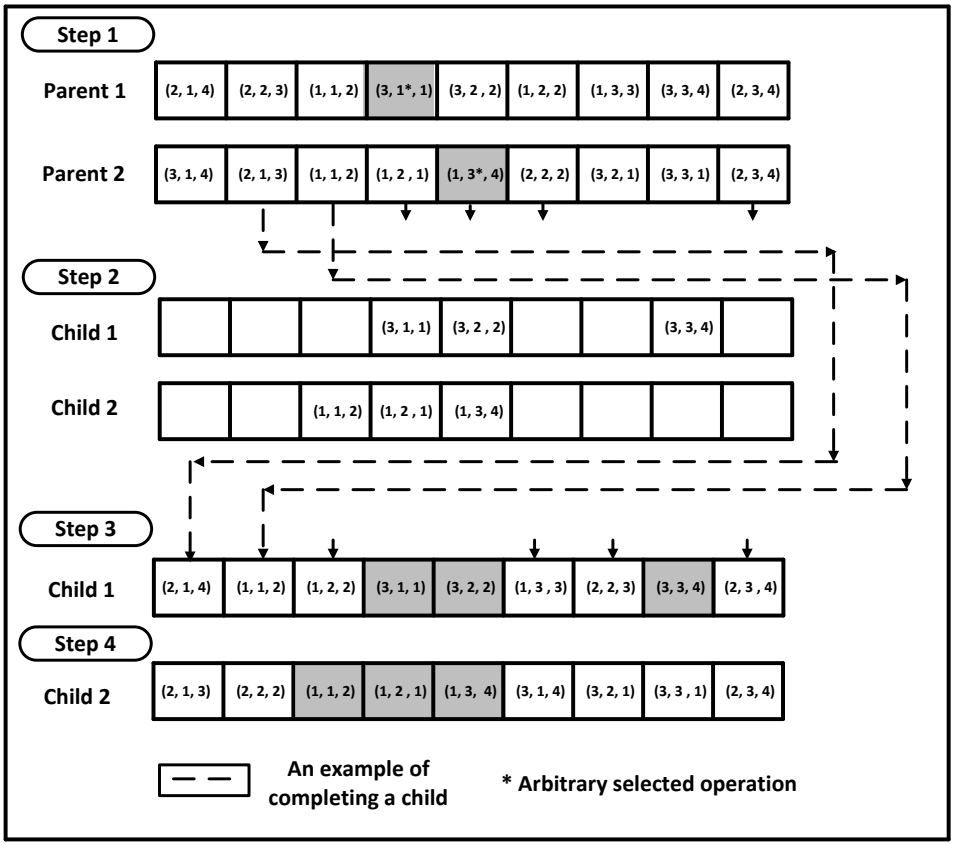


Figure 7.3: Sequencing crossover

Unlike the crossover operators, the mutation operators applied on each chromosome to reverse a selected chromosome bit pattern. By this concept, lost or

disturbing genetic information is recovered. In contrast to crossovers aim to exploit the current solution to create better fitted ones, mutations are considered to assist whole solution space exploration. We propose Assignment mutation to alter few allele m of a given individual chromosome in probability ρ_3. Second mutation operator, intelligent mutation, is used to randomly select an operation on the machine with maximum workload and assign it to a eligible machine with minimum workload. This mutation is applied with ρ_4. Last mutation operator is based on precedence preserving shift mutation (PPS) operators of (Lee *et al.*, 1998). PPS choose an operation from individual parent and moves it into another position while respecting of the precedence constraints for that operation. Figure shows an example of applying PPS operator. This operator is applied with a small probability ρ_5.

Step 1. Set $l = 1$

Step 2. Set the values of indices j, o and m as obtained from the gene at location l of the chromosome under consideration.

Step 3. Calculate the completion time $c_{o,j,m}$

7.3.3. Fitness Evaluation

The fitness of a solution is calculated by combining the two objectives into a weighted sum. As noted in Section 6.4, the objective values on the two criteria have to be normalized before they are summed because they are of different range and magnitudes. Let $C_M(s)$ be the makespan of the s^{th} chromosome. The scaled makespan $C_M^{trans}(s)$ of a solution s is as follows:

$$C_M^{trans}(s) = \begin{cases} \frac{C_M(s) - C_M^{min}(s)}{C_M^{max}(s) - C_M^{min}(s)} & ; \text{ If } \quad C_M(s) \neq C_M^{min}(s) \\ 0 & ; \text{ Otherwise} \end{cases} \tag{7.16}$$

where $C_M^{min}(s) = min\{C_M(s)\}; \quad C_M^{max}(s) = max\{C_M(s)\}; \quad \text{For all} \quad 1 \leq s \leq K$ (7.17)

Where K is the total number of solution candidates to be evaluated in a generation. With the same method, we can normalize total traveling distance for each solution s,as given in Eqs. (7.18) and (7.19).

$$MHCost^{trans}(s) = \begin{cases} \frac{MHCost(s) \; - \; MHCost^{min}(s)}{MHCost^{max}(s) \; - \; MHCost^{min}(s)} & ; \text{ If } \quad MHCost(s) \neq MHCost^{min}(s) \\ 0 & ; \text{ Otherwise} \end{cases} \tag{7.18}$$

where $MHCost^{min}(s) = min\{MHCost(s)\}; \quad MHCost^{max}(s) = max\{MHCost(s)\};$ For all $1 \leq s \leq K$. (7.19)

During scaling fitness value for genes within a population, it is possible that both $MHCost^{trans}$ and $C_M^{trans}(s)$ equal to zero for a particular gene and that gives objective function of zero. This situation arises where $MHCost(s) = MHCost^{min}$ and $C_M(s) = C_M^{min}(s)$ for a particular gene. This individual can be excluded from the existing population and included to a new population will create in the next generation. By doing that, one can maintain the individual solution with good quality in search process and also avoid algorithm stagnation.

After scaling, the two objectives all take values from the range of $[0, 1]$. In order to guide the genetic and local search to the most promising area of search

space, makespan is given somewhat more weight because it is typically the most important criterion in practical production environments.

7.3.4. Determination of the Staring and Completion Times of Jobs

In order to determine the starting and completion times of jobs for an individual solution, one need to obtain assignment and sequencing information from the chromosome. By doing it, a makespan can be calculated and used as a measure of this chromosome fitness. To properly calculate the makespan, all attributes incorporated in the model need to be taken into account. These are sequence-dependent setup, machine release date, lag time and the nature of the setup (anticipatory and non-anticipatory). The decoding procedure is consisted of five steps based on the sequencing and assignment information obtained from gene of the chromosome under consideration.

Step 1. Set $l = 1$

Step 2. Set the values of indices j, o and m as obtained from the gene at location l of the chromosome under consideration.

Step 3. Calculate the completion time $c_{o,j,m}$

- If (1) operation o of job j is the first operation assigned on machine m and (2) $o = 1$, then $c_{o,j,m} = D_m + S^*_{o,j,m} + B_j \cdot T_{o,j,m}$.
- If (1) operation o of job j is the first operation assigned on machine m, (2) $o > 1$, and (3) operation o-1 is assigned on machine m', then $c_{o,j,m} = \max\{D_m + (1 - A_{o,j}) \cdot S^*_{o,j,m};\quad c_{o-1,j,m'} + L_{o,j}\} + B_j \times T_{o,j,m} + A_{o,j} \cdot S^*_{o,j,m}$.
- If (1) operation o' of job j' is the operation to be processed immediately before operation o of job j on machine m and (2) $o = 1$,

then $c_{o,j,m} = c_{o',j',m} + S_{o,j,m,o',j'} + B_j \cdot T_{o,j,m}$.

- If (1) operation o' of job j' is the operation to be processed immediately before operation o of job j on machine m, (2) $o > 1$, and (3) operation o-1 is assigned on machine m', then $c_{o,j,m} = \max\{c_{o',j',m} + (1 - A_{o,j}) \cdot S_{o,j,m,o',j'};\quad c_{o-1,j,m'} + L_{o,j}\} + B_j \cdot T_{o,j,m} + A_{o,j} \cdot S_{o,j,m,o',j'}$.

Step 4. If l is less than the total number of operations, increase its value by 1 and go to Step 2; otherwise go to Step 5

Step 5. Calculate the makespan of the schedule as $c_{max} = \max\{c_{o,j,m};\ \forall(o,j,m)\}$ and set the fitness of the individual under consideration to c_{max}.

Chapter 8

Numerical Examples: Scheduling

Since literature does not yet exist for an integrated model that considers simultaneously machine assignment and scheduling in manufacturing systems with distributed layouts, there is no comparable data for us to use. Therefore, we generated several data sets to illustrate the problem and demonstrate the performance of the proposed solution procedure. A small problem instance ($id : Small20 \times 22$) consisting of the processing of 25 jobs in a manufacturing with a distributed layout comprised of 22 machines. The processing data for Problem $id : Small20 \times 22$ is given in Table 8.1 and Table 8.3. Table 8.1 contains the batch size, material handling cost per unit distance for one unit of job j, the number of operations for each job, the nature of setup (attached or detached) and lag time. Similar to Problem 1 which is given in Section 5.1, we considered a system composed of 20 resource elements and 22 machine tools with four different levels of sharing processing capabilities (REs) which is shown in Table 8.2. Recall that Case 1 represents a situation in which a particular RE is available on several machine tools; Case 4 represents a situation where most of the machines have unique capabilities; and Cases 2 and 3 lie in between the two extremes. Table 8.3 provides indices of the alternative resource elements (REs) available in machines, and corresponding processing times for a layout with Case 1.

Table 8.1: Lag time and the nature of setup for the parts in Problem *id* : $Small20 \times 22$

			Operation data $(A_{o,j}, L_{o,j})$				
Job	B_j	F	o = 1	o = 2	o = 3	o = 4	o = 5
1	210	2	(1, 0)	(0, 10)			
2	200	3	(1, 0)	(1, 10)	(1, 10)		
3	170	2	(1, 0)	(0, 90)	(1, 10)	(1, 10)	
4	220	1	(1, 0)	(1, 10)			
5	160	2	(1, 0)	(1, 10)			
6	200	2	(1, 0)	(1, 10)			
7	180	2	(1, 0)	(0, 10)	(1, 90)	(0, 10)	
8	150	1	(1, 0)	(1, 10)	(1, 10)	(1, 10)	(1, 10)
9	160	3	(1, 0)	(0, 10)	(1, 10)	(1, 10)	(1, 10)
10	170	1	(1, 0)	(1, 10)	(0, 10)	(1, 10)	
11	190	3	(1, 0)	(1, 10)	(1, 10)		
12	140	2	(1, 0)	(0, 10)	(0, 10)		
13	230	3	(1, 0)	(1, 10)	(1, 90)		
14	210	2	(1, 0)	(1, 10)	(1, 10)	(1, 90)	
15	210	3	(1, 0)	(1, 10)	(1, 10)	(1, 90)	
16	160	2	(1, 0)	(1, 10)	(1, 10)	(1, 90)	
17	180	3	(1, 0)	(1, 50)	(1, 10)		
18	210	3	(1, 0)	(1, 10)			
19	150	1	(1, 0)	(1, 10)	(0, 10)	(1, 90)	(1, 10)
20	180	2	(1, 0)	(1, 10)	(1, 10)		
21	140	2	(1, 0)	(1, 10)	(1, 90)	(1, 10)	
22	160	3	(1, 0)	(0, 10)	(1, 10)	(1, 10)	
23	190	1	(0, 0)	(0, 10)	(1, 10)	(0, 10)	
24	210	3	(0, 0)	(0, 10)	(0, 10)	(1, 10)	
25	200	2	(1, 0)	(1, 10)	(1, 10)		

We also proposed an optimized distributed layout depicted in Figure 8.1. The machine configuration is obtained from solving the Problem *id* : $Small20 \times 22$ with Case 1 using the developed model and solution procedure in Part I of this thesis. The material handling distances between each pair of locations are shown in Table A.6 in the Appendix **??**.

Table 8.2: Resource Elements data in Problem $id : Small20 \times 22$

Resource Element r	Indices of machines having resource element r			
	Case 1	Case 2	Case 3	Case 4
1	(1, 2, 3, 4, 5, 6)	(1, 2, 4, 6)	(1, 6)	(1)
2	(1, 2, 3, 4, 5, 6)	(1, 3, 5)	(2, 5)	(2)
3	(1, 2, 3, 4, 5, 6)	(2, 4, 6)	(3)	(3)
4	(1, 2, 3, 4, 5, 6)	(1, 3, 5, 6)	(4)	(4)
5	(7, 8, 9, 10)	(7, 8, 10)	(8)	(5)
6	(7, 8, 9, 10)	(9, 10)	(7, 10)	(6)
7	(7, 8, 9, 10)	(7, 8, 10)	(9)	(7)
8	(7, 8, 9, 10)	(8, 9)	(7, 10)	(8)
9	(11, 12, 13, 14, 15, 16)	(11, 14, 15)	(14, 15)	(9)
10	(11, 12, 13, 14, 15, 16)	(11, 13, 16)	(11, 16)	(10)
11	(11, 12, 13, 14, 15, 16)	(12, 14, 16)	(16)	(11)
12	(11, 12, 13, 14, 15, 16)	(11, 13, 15)	(12, 13)	(12)
13	(11, 12, 13, 14, 15, 16)	(11, 12, 14)	(11, 15, 16)	(13)
14	(17, 18, 19)	(18, 19)	(18)	(14)
15	(17, 18, 19)	(17, 19)	(17)	(15)
16	(17, 18, 19)	(18, 19)	(19)	(16)
17	(20, 21, 22)	(20, 22)	(22)	(17)
18	(20, 21, 22)	(21, 22)	(20)	(18, 21)
19	(20, 21, 22)	(20, 21)	(21)	(19, 22)
20	(20, 21, 22)	(20, 22)	(20, 22)	(20)

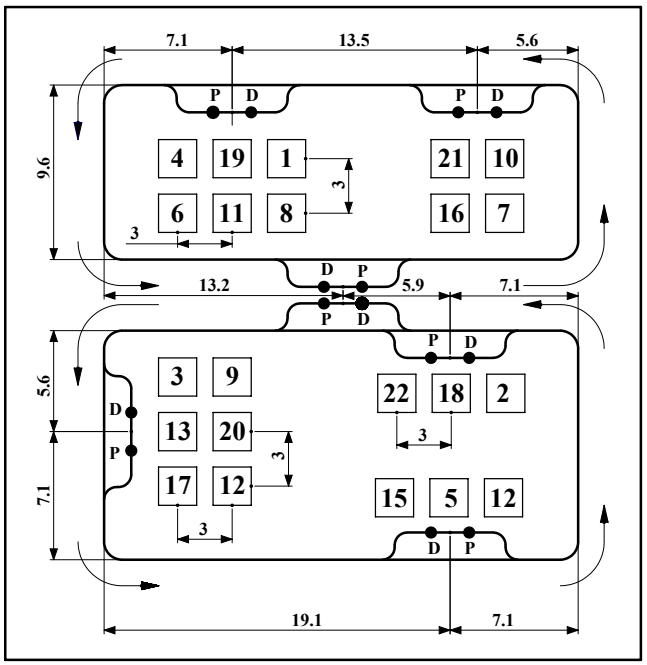


Figure 8.1: An optimized distributed layout obtained from solving Problem id : $Small20 \times 22$ - dimensions are in unit distance

Table 8.3: Alternative routes and processing time for the parts in Problem $id : Small20 \times 22$

	Alternative routes (r), Processing time ($T_{o,j}$)				
Job	o = 1	o = 2	o = 3	o = 4	o = 5
1	(17, 18, 19), (1)	(11, 12, 13, 14, 15, 16), (1)			
2	(20, 21, 22), (2)	(20, 21, 22), (1)	(1, 2, 3, 4, 5, 6), (2)		
3	(7, 8, 9, 10), (2)	(11, 12, 13, 14, 15, 16), (1)	(7, 8,9, 10), (2)	(1, 2, 3, 4, 5, 6), (3)	
4	(17, 18, 19), (1)	(20, 21, 22), (2)			
5	(7, 8, 9, 10), (2)	(17, 18, 19), (1)			
6	(17, 18, 19), (3)	(17, 18, 19), (2)			
7	(1, 2, 3, 4, 5, 6), (2)	(1, 2, 3, 4, 5, 6), (1)	(20, 21, 22), (3)	(1, 2, 3, 4, 5, 6), (1)	
8	(1, 2, 3, 4, 5, 6), (2)	(11, 12, 13, 14, 15, 16), (1)	(17, 18, 19), (3)	(1, 2, 3, 4, 5, 6), (2)	(7, 8, 9, 10), (2)
9	(7, 8, 9, 10), (2)	(7, 8, 9, 10), (2)	(11, 12, 13, 14, 15, 16), (2)	(1, 2, 3, 4, 5, 6), (2)	(20, 21, 22), (2)
10	(20, 21, 22), (2)	(7, 8, 9, 10),(1)	(20, 21, 22), (1)	(20, 21, 22), (2)	
11	(11, 12, 13, 14, 15, 16), (1)	, (11, 12, 13, 14, 15, 16), (1)	(1, 2, 3, 4, 5, 6), (3)		
12	(7, 8, 9, 10), (3)	(17, 18, 19), (1)	(7, 8, 9, 10), (2)		
13	(11, 12, 13, 14, 15, 16), (2)	(11, 12, 13, 14, 15, 16), (2)	(11, 12, 13, 14, 15, 16), (2)		
14	(20, 21, 22), (3)	(17, 18, 19), (2)	(7, 8, 9, 10), (2)	(17, 18, 19), (1)	
15	(20, 21, 22), (2)	(7, 8, 9, 10), (1)	(17, 18, 19), (2)	(7, 8, 9, 10), (2)	
16	(1, 2, 3, 4, 5, 6), (2)	(11, 12, 13, 14, 15, 16), (2)	(17, 18, 19), (2)	(1, 2, 3, 4, 5, 6), (3)	
17	(20, 21, 22), (2)	(11, 12, 13, 14, 15, 16), (3)	(1, 2, 3, 4, 5, 6), (2)		
18	(20, 21, 22), (2)	(11, 12, 13, 14, 15, 16), (2)			
19	(1, 2, 3, 4, 5, 6), (2)	(20, 21, 22), (1)	(17, 18, 19), (3)	(7, 8, 9, 10), (1)	(1, 2, 3, 4, 5, 6), (1)
20	(20, 21, 22), (2)	(11, 12, 13, 14, 15, 16), (2)	(7, 8, 9, 10), (2)		
21	(11, 12, 13, 14, 15, 16), (2)	(7, 8, 9, 10), (2)	(7, 8, 9, 10), (2)	(20, 21, 22), (2)	
22	(17, 18, 19), (3)	(20, 21, 22), (2)	(7, 8, 9, 10), (2)	(11, 12, 13, 14, 15, 16), (3)	
23	(20, 21, 22), (2)	(20, 21, 22), (1)	(7, 8, 9, 10), (2)	(11, 12, 13, 14, 15, 16), (1)	
24	(7, 8, 9, 10), (1)	(11, 12, 13, 14, 15, 16), (3)	(7, 8, 9, 10), (2)	(7, 8, 9, 10), (2)	
25	(17, 18, 19), (3)	(7, 8, 9, 10), (2)	(11, 12, 13, 14, 15, 16), (3)		

In our experiments, we also consider two others layout sizes having 45 and 75 machines with 34 and 56 resource elements (Problem $id : Mediume45 \times 34$ and Problem $id : Large75 \times 56$).

In order to examine the impact of transportation time and proposed algorithm's efficiency, several numbers of jobs and maximum number of operation for each job were considered for both size of above mentioned layouts.
Multi Objective Genetic Algorithm described in Section 7.3 was coded in C++ programming language using MPI message-passing library for communication. The code was tested in a parallel computation environment composed of 872 waiters containing each one Pentium 4 processor (3.2 GHz, 2GB RAM).

We conducted numerical experiments to answer the following three questions:

1. How can decision makers trade-off between makespan and martial handling cost?

2. How much cost saving can be realized by introducing material handling cost in scheduling of a system with distributed layout?

3. Is that desirable to explicitly incorporate transportation time in the model or can it influence the completion time of the jobs?

4. How the levels of sharing processing capabilities (Case1 to 4) can affect the cost saving resulting from incorporating material handling cost in the scheduling?

5. How good is the quality of the solution we obtain, and how is it affected by problem characteristics, such as number of jobs and maximum number of operation for each job?

8.1. Trade-off between Makespan and Martial Handling Cost

Figure 8.2 quantifies the trade-off between makespan performance and material handling performance in solving Problem $id : Small20 \times 22$ for various weighting parameter (α) from 0 to 1 in the interval of 0.1. When $\alpha = 1$, only the makespan objective is considered and when $\alpha = 0$, only the material handling cost objective is accounted. It is obvious that the makespan decreases when the weighting parameter increases from 0 to 1. However, material handling cost (labeled MH Cost) increases much more slowly when weighting parameter increase from from 0 to 1. This means that for higher alpha values, the shorter makespan obtained, while the material handling performance advantage tends to disappear. Therefore, the choice of a scheduling policy is a key decision that greatly affects model's ability to meet pragmatic constraints. Two curves meet each other at one point where two conflicting objective functions are minimised. No other solution has better value than this solution for makespan and is not worse than the solution for the material handling cost. This means that the solution is global Pareto optimal.

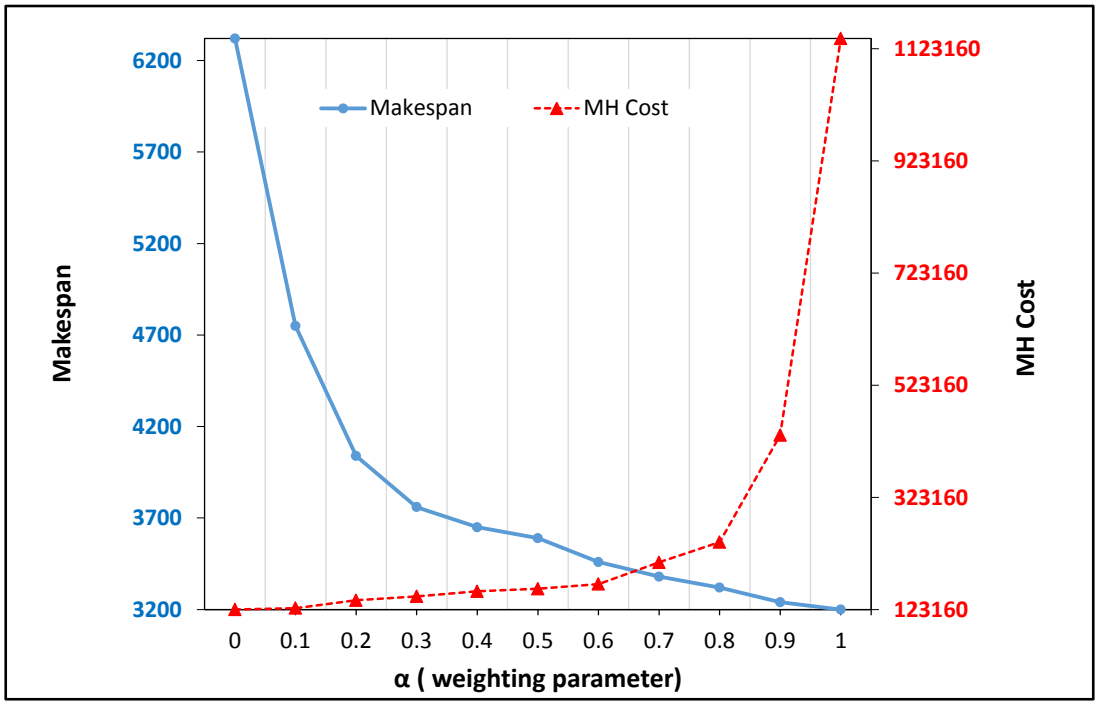


Figure 8.2: Trade-off between makespan and martial handling cost in Problem $id : Small20 \times 22$

Recall that the solutions that are non-dominated within the entire search space are denoted as Pareto optimal solution and constitute the Pareto optimal set. This set is also known as Pareto optimal front. Pareto optimal front in 11 separate runs for Problem $id : Small20 \times 22$ is also shown in Figure 8.3.

8.2. The Effect of Overlapping Capabilities

In order to examine the role of levels of sharing processing capabilities on material handing cost in optimized schedule, Problem $id : Small20 \times 22$ with four different Cases and 25 parts are considered. The weights used for combining multiple objectives into a scalar fitness function (α) are arbitrarily selected such that $0 \leq \alpha \leq 1$. Figure 8.4 indicates that material handling cost is optimized in greater

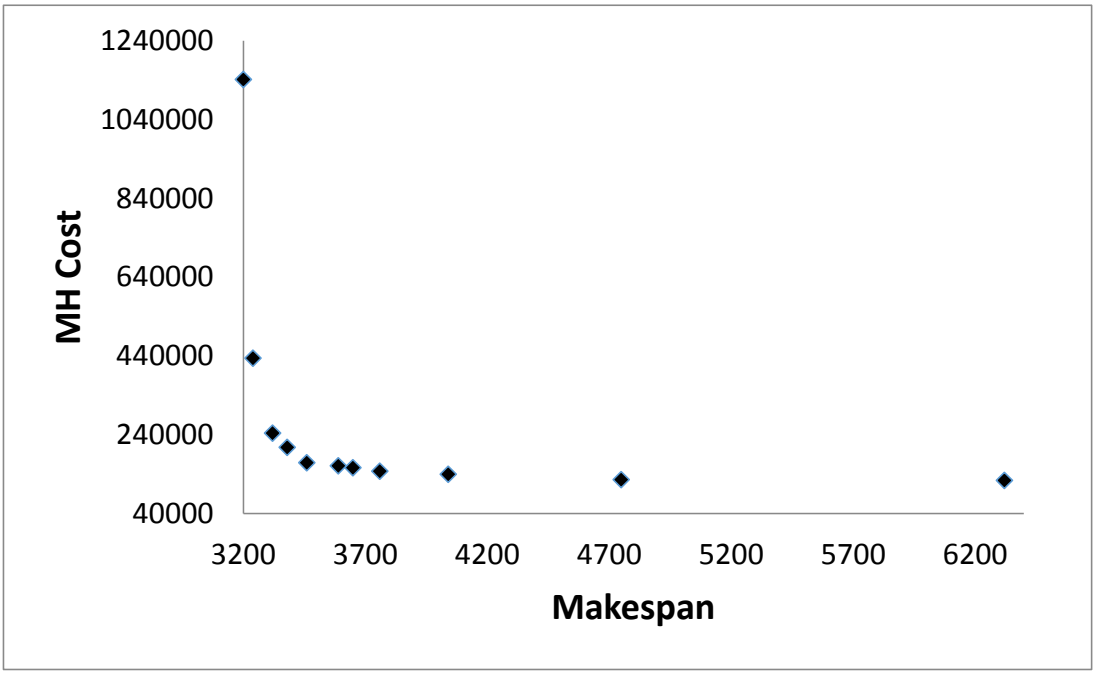


Figure 8.3: Pareto optimal front for Problem $id : Small20 \times 22$ with various (α) from 0 to 1

extend in case 1 and 2 compared to case 3 and 4. The obtained result supports the idea that Case 1 and 2 representing systems with more highly overlapping machine capabilities have more potential to be optimized than other Cases.

8.3. Scheduling in Distributed versus Functional Layout

The aim of this Section is to illustrate the greater effectiveness of incorporating material handling cost in scheduling of a manufacturing system with distributed layout compared with functional layout where machines are aggregated in shops by the nature of skills and technological processes involved. The Problem $id : Small20 \times 22$ with Case 1 was solved to optimize simultaneously both

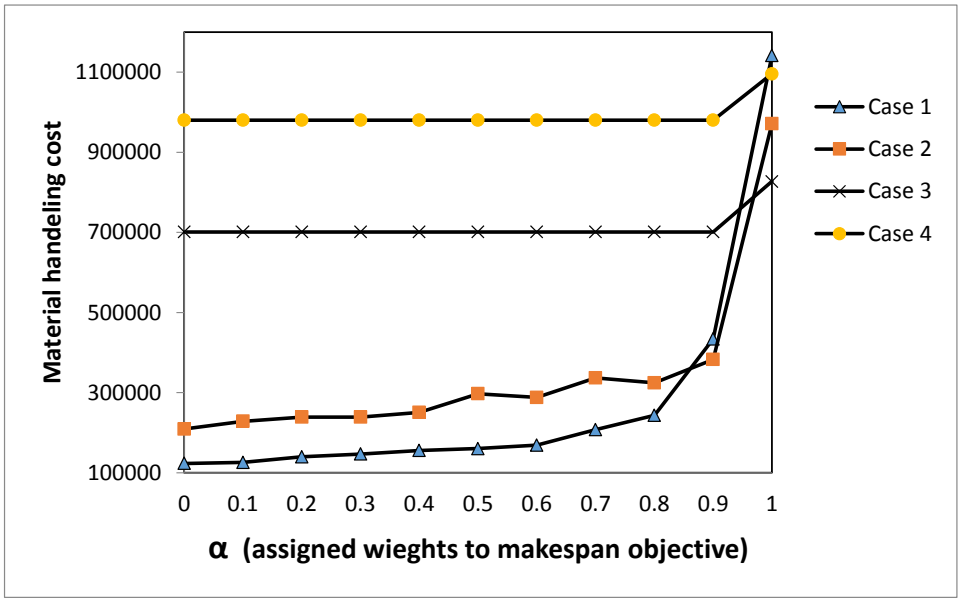


Figure 8.4: A comparison of material handelig cost for all Cases 1-4 in optimized solution

makespan and material handling cost. With assigning a set of weight values (α) ranging between 0 and 1 to makespan objective, several optimized solution were obtained such that material handling cost optimized ingreater extend while weight value decreases from 1 to 0. Figure 8.5 shows that material handling cost drastically decreases in distributed layout scheduling with assigning α from 1 to 0. In contrast, in the functional layout scheduling, optimization leads to mild decrease in material handling cost with a same assigned set of weight values. This difference is mainly related to separation of machines in shop floor in distributed layout and thus increases total traveling distance. The result shows the importance of incorporating material handling cost into objective function in order to simultaneously find optimum allocation of the jobs to machine and starting times of the job on each machine in distrusted shop environment compared to functional one.

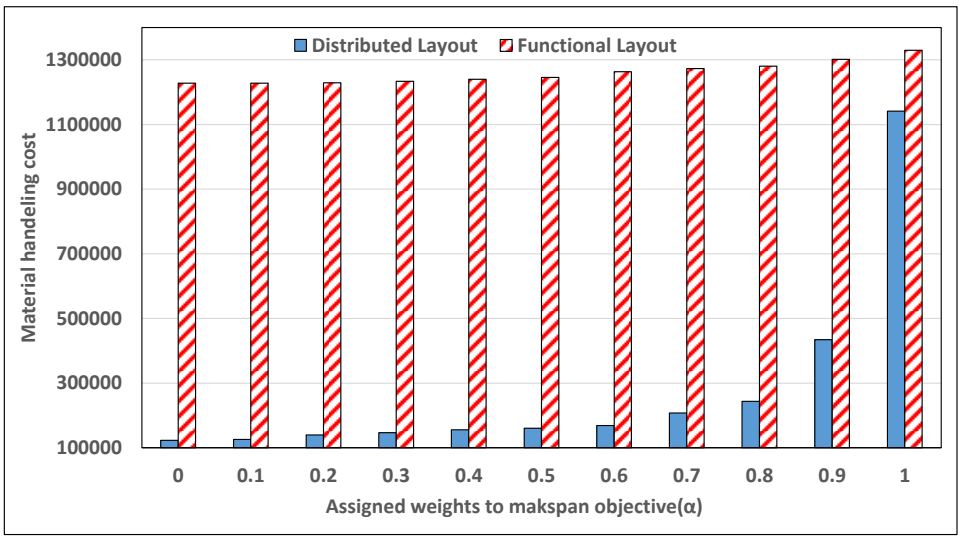


Figure 8.5: A comparison of material handelig cost reduction in distributed and functional layouts

8.4. The Effect of Transportation Time

In order to examine the influence of time taken to transport jobs on makespan, we solved Problems $id : Small20 \times 22$ and $id : Mediume45 \times 34$ with 25 and 50 parts respectively. In this experiment, we also considered different average transportation time for all jobs involving in the system. This means the transportation time is not job dependent. In this situation, transportation time was modeled simply as a minimum time lag with the average time it takes to carry out jobs between machines.

Figure 8.6 reveals that transportation time has a limited influence on makespan in both size problems because average transportation time is increased by 50 times and the makespan is remained remarkably unchanged. The main reason might be that machines are busy and have queues of jobs before them at almost all time in this type of multi-product manufacturing system. Efficient utilization of resource in modern manufacturing system doesn't allow to machines become free.

As shown in Figure 8.7, the optimized schedule indicates that machine utilization is high and the shop is almost fully loaded.

In fact, transportation time can be added to the job waiting time in queues. As result, it only very slightly affects makespan. The results obtained in this Section prove that why the proposed model has not explicitly considered transportation time, because optimized makespan is largely independent of transportation time.

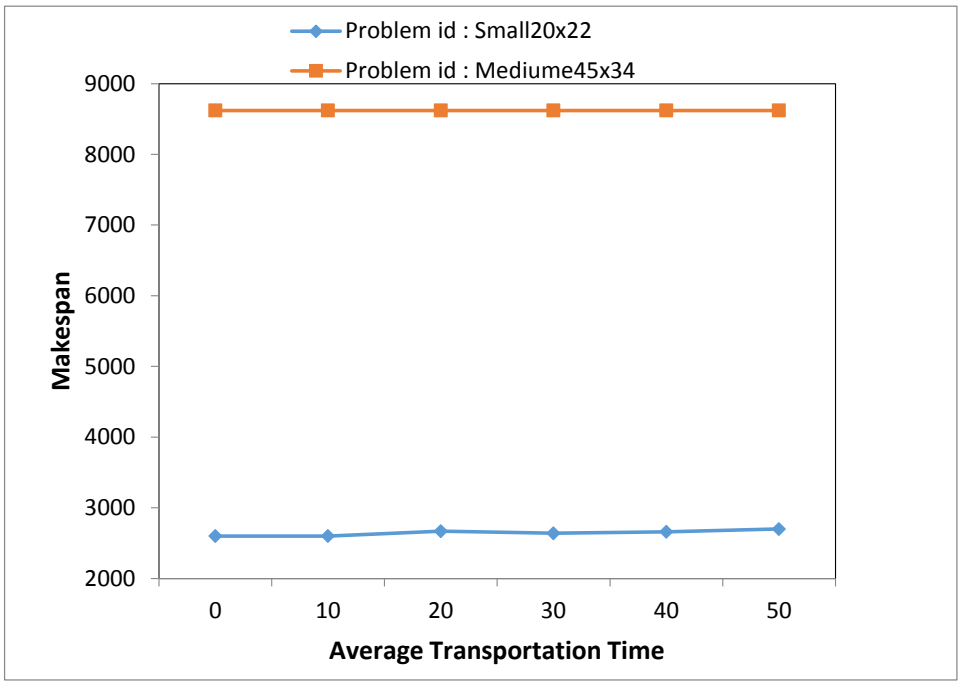


Figure 8.6: The influence of time taken to transport jobs on makspan

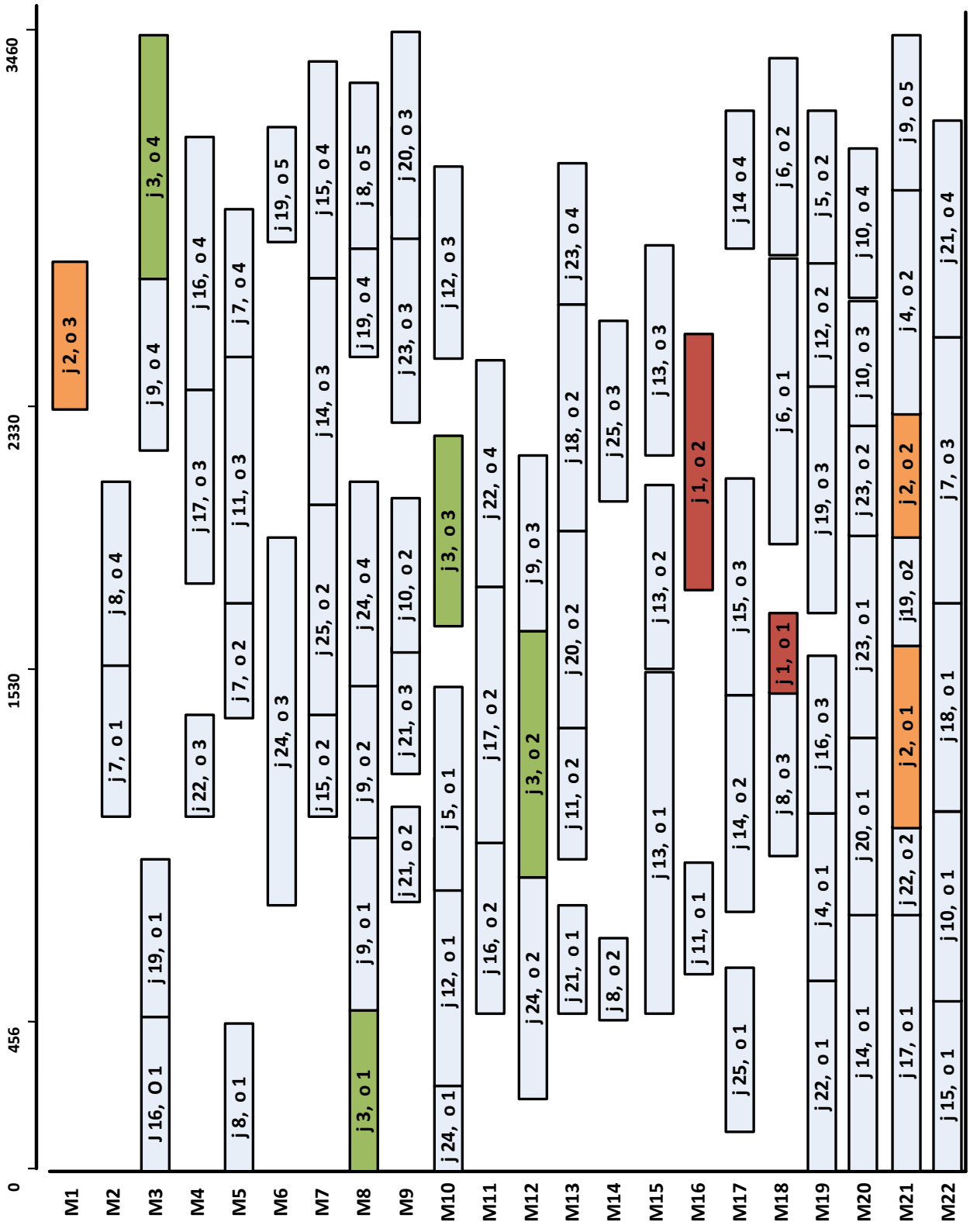


Figure 8.7: Gantt chart obtained by solving Problem $id : Small20 \times 22$ with $\alpha = 0.6$

Chapter 9

Conclusions and Future Research

9.1. Conclusions

In today's increasingly competitive and demanding marketplace, a high-performance production, responsiveness, efficiency and flexibility are challenges.
Research in design of layouts for multi-product enterprises working in dynamic environment has been conducted extensively, however, only a few publications have addressed these changes using a comprehensive model to design a distributed layout. A comprehensive design methodology is essential since the facility layout, material handling system, process routings and production plan must all fit together to enable a competitive manufacturing performance. Another aspect which is rarely addressed when designing distributed layouts is the optimization of several cost elements in an integrated manner instead of trying to consider only material handling cost. Thus, several important design challenges need to be addressed in order to maximize the benefits of layouts design.

The first goal of this research was to develop a design methodology which addresses the challenges of meeting high operational efficiency and flexibility in highly volatile environments. In the first part of research, we developed a new mathematical model that integrates layout configuration and production planning

in the design of dynamic distributed layouts. The model incorporates a number of important manufacturing attributes such as demand fluctuation, system reconfiguration, lot splitting, work load balancing, alternative routings, machine capability and tooling requirements. In addition, the model allows the optimization of several cost elements in an integrated manner. These include material handling, machine relocation, setup, inventory carrying, in-house production and subcontracting costs. However, optimal solutions for the proposed mathematical model can only be found for small size problems due to NP-complexity. This leds to the development of three heuristic methods which can handle large distributed layout problems in a reasonable amount of time. These heuristics are as follows:

- Linear programming embedded simulated annealing algorithm (LPSA)
- Linear programming embedded genetic algorithm (LPGA)
- Pure simulated annealing algorithm (PSA)

Although the solution representation encoding integer variables of solution is identical in both LPSA and LPGA and they use a same approach to hybridize a metaheuristic with linear programming, their metaheuristic parts are incredibly diverse in nature. Such fact motivated us to develop and investigate performance of these solution procedures.

Since the comprehensive problem addressed in this study has not been previously presented, we had no comparable examples from the literature to use. Therefore, in order to illustrate the considered problem and demonstrate the performance of the proposed solution procedures, we generated several data sets. The experimental study was designed to compare distributed layout with functional layout and investigate some design factors affecting the performance of distributed layouts, namely, level of overlapping capabilities of machines, dynamic configuration, workload balancing, production planning and subcontracting. Several insights into distributed layout design can be stated as follows.

1. Using distributed layouts results in significant savings in material handling cost as important cost component within manufacturing.

2. Distributed layouts are highly desirable in a situation where there are many machine tools with several shared capabilities.

3. Dynamic reconfiguration can bring a significant cost savings when the manufacturing system has more unique machines with less shared capabilities.

4. Including several pragmatic issues of the manufacturing system can significantly effects manufacturing cost.

The performance of three heuristic methodologies was evaluated via a comparative study from a solution convergence, solution quality and algorithm robustness point of view. Overall, the heuristics performed well under different circumstances. The results from the experimental study can be summarized as follows.

1. Both LPSA and LPGA perform well under different circumstances.

2. The PSA is not capable of solving the model considering a workload balancing constraint since it tries to address workload balancing using a penalty method which is an indirect way of constraint handling.

3. Both LPSA and LPGA outperform PSA in terms solution quality, solution convergence and implementation simplicity.

4. LPGA is best with respect to solution convergence, solution quality and algorithm robustness point of view. It reflects the fact that, Genetic algorithms are superior to simulated annealing algorithms in covering a much larger landscape of the search space at each iteration because GA uses a population based selection while SA utilizes one point in each iteration.

The second goal of this research was to develop a comprehensive scheduling model to determine an optimum schedule for a manufacturing system with distributed

layout in a way that total transportation cost of parts and makespan are minimized. To our limited knowledge, scheduling in manufacturing system with distributed layouts has not been addressed in the literature. Using a multi-objective mixed integer programming model, machine assignment and scheduling are simultaneously optimized. In other words, one goal of the problem is to find a schedule of operations on machines (starting time of operation) which minimize the overall finishing time or makespan. Another one is to find assignment of jobs to the machines such that total distance traveled by parts is minimized. Therefore, selected objectives are essentially limited to those which are related the nature of manufacturing with distributed layout. Most importantly, the model incorporating sequence-dependent setup time, attached or detached setup time, machine release dates, and time lag requirements because there is clearly need to consider an integrated scheduling model to address multi-faceted nature of the real world.

However, mathematical models for scheduling problems are normally much more difficult to solve due to the nature of combinatorial optimization. In order to efficiently solve the developed model, we proposed a parallel genetic algorithm that runs on a parallel computing platform. Since the comprehensive scheduling model has not been previously presented, we have no comparable examples from the literature to use. Therefore, in order to examine the model and evaluate the performance of heuristic, we generated several data sets and scenarios that take into account several numbers of machines and parts, maximum number of operation per each part, and levels of overlapping capability. We first compared the results from the scheduling model in two different shop environment, distributed and functional layouts, to observe material handling costs which is explicitly captured in the model. We also focused on the two system characteristic on the schedule solution: (1) the level of overlapping machine capabilities and (2) the average transportation time.

Our analysis provides several insights into scheduling in distributed layout which can be stated as follows.

1. An additional cost saving can be realized by incorporating material handling cost in scheduling objective optimization of a system with distributed layout. We achieved average total martial handling cost savings of over 40% relative to scheduling in system with functional layout.

2. It is not desirable to explicitly incorporate transportation time in the model since it does influence the completion time of the jobs due to nature of manufacturing system.

3. The martial handling cost saving resulting from a system with less shared process capability was lesser than system with highly overlapping capabilities due to limited number alternative jobs routings.

The performance of the parallel GA approach was evaluated against a sequential GA. The algorithm demonstrates substantial reductions of computing time and improves the search.

9.2. Contributions

The major contributions of our research lie in the following aspects:

1. We formulated mixed linear integer program that accurately captures the design of dynamic distributed layouts in a comprehensive manner. The distributed layout represents an attempt to achieve an aggregation of machines through the facility in meaningful manner. Thus, in contrast to traditional facility, the layout can address changes in the production environment and adapt to volatilities.

2. Three new heuristics for solving dynamic distributed layouts problems in a reasonable amount of time was developed. We developed two hybrid approaches that exploit the structure of the formulation using a heuristic procedure which embeds linear programming and follows multiple search paths. These algorithms provide near-optimal solutions in many instances.

3. We achieved average total distributed layout cost savings of over 55% relative to functional layout. That is, we can substantiate distributed layouts are highly desirable especially in situation where there are many machine tools with several shared capabilities. This is because when there are several machine tools with several shared capabilities, the distribution of these machine tools makes these capabilities easily accessible from different regions of the layout thereby reducing material handling cost significantly.

4. In this work, also a multi-objective mixed integer programming model for scheduling of manufacturing systems with distributed layouts was developed such that machine assignment and scheduling are simultaneously examined. By incorporating the material handling cost into objective function, we were able to show that a significant martial handling cost saving can be obtained while maintain an appropriate schedule for production. Refining our analysis, we examined several data sets and determined that it is particularly beneficial for a system comprised of machines with higher overlapping of capabilities. It validates and extends results presented in first part of our work that prove distributed layouts are highly desirable in a situation where there are many machine tools with several shared capabilities.

5. A novel multi-objective genetic algorithm for solving the scheduling in system with distributed layouts problem was designed. Rather than considering sequential GA requires considerable size of computer memory when the size of population needs to be very large, we followed a parallel GA

using high performance parallel computing for this type of large and complex problem. The parallel algorithm approach demonstrates substantial reductions of computing time and improves the search performances.

9.3. Future Research

This section proposes several directions for future research on design of distributed layouts, which are described as follow:

1. One area of future research in the distributed layout design problem includes the development of more effective and efficient solution procedures, particularly for analysing large problem instances, i.e. more than 150 parts with maximum number of operation equal to 50. Our limitation on access to a parallel computing platform where there are linear programming solvers for each processor make it necessary to use extended solution procedure for further investigation.

2. We believe that the comprehensive distributed layout problem presented in this study raises many interesting questions regarding machine aggregation and part flow allocation that need further exploration.

3. One possibility would be to include some material handling aspects into the facility design method. Clearly, there is a close link to layout and material handling design. Especially when consecutive operations can be conveyed on machines those are close to each other.

4. In the future, additional studies should be designed that simulate both the facility design and scheduling models in order to investigate their performance and to illustrate how the methods are applied to a real data set.

5. In the facility layout design model, due to the complex nature of the problem, there was so difficult to consider stochastic demand fluctuation, lot

streaming, machines dimensions, closeness relationships and tools consumption cost however, these constraints can be studied in the future researches.

Bibliography

Abdelmaguid, T. F., Nassef, A. O., Kamal, B. A., and Hassan, M. F., 2004. A hybrid ga/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International journal of production research*, **42 (2)**, 267–281.

Agrawal, G. K. and Heragu, S. S., 2006. A survey of automated material handling systems in 300-mm semiconductorfabs. *Semiconductor Manufacturing, IEEE Transactions on*, **19 (1)**, 112–120.

Arkat, J., Farahani, M. H., and Ahmadizar, F., 2012. Multi-objective genetic algorithm for cell formation problem considering cellular layout and operations scheduling. *International Journal of Computer Integrated Manufacturing*, **25 (7)**, 625–635.

Arvindh, B. and Irani, S., 1994. Cell formation: the need for an integrated solution of the subproblems. *International Journal of Production Research*, **32 (5)**, 1197–1218.

Asef-Vaziri, A. and Laporte, G., 2005. Loop based facility planning and maerial handling. *European Journal of Operational Research*, **164 (1)**, 1–11.

Askin, R., Ciarallo, F., and Lundgren, N., 1999. An empirical evaluation of holonic and fractal layouts. *International Journal of Production Research*, **37 (5)**, 961–978.

Baker, K. R., 1995. Lot streaming in the two-machine flow shop with setup times. *Annals of Operations Research*, **57 (1)**, 1–11.

Balakrishnan, J. and Cheng, C. H., 1998. Dynamic layout algorithms: a state-of-the-art survey. *Omega*, **26 (4)**, 507–521.

Balakrishnan, J. and Cheng, C. H., 2007. Multi-period planning and uncertainty issues in cellular manufacturing: A review and future directions. *European Journal of Operational Research*, **177 (1)**, 281–309.

Bayat Movahed, S., 2014. Linear programming assisted genetic algorithm for solving a comprehensive job shop lot streaming problem. Master's thesis.

Baykasoglu, A., 2003. Capability-based distributed layout approach for virtual manufacturing cells. *International Journal of Production Research*, **41 (11)**, 2597–2618.

Benajaafar, S., 1995. Design of flexible layouts for manufacturing systems. In: Engineering Management Conference, 1995. Global Engineering Management: Emerging Trends in the Asia Pacific., Proceedings of 1995 IEEE Annual International. IEEE, pp. 421–427.

Benjaafar, S., Heragu, S. S., and Irani, S. A., 2002. Next generation factory layouts: Research challenges and recent progress. *Interfaces*, **32**, 58–77.

Benjaafar, S. and Sheikhzadeh, S., 2000. Design of flexible plant layouts. *IIE Transactions*, **32**, 309–322.

Bilge, Ü. and Ulusoy, G., 1995. A time window approach to simultaneous scheduling of machines and material handling system in an fms. *Operations Research*, **43 (6)**, 1058–1070.

Blum, C., Puchinger, J., Raidl, G. R., and Roli, A., 2006. Hybrid metaheuristics in combinatorial optimization: A survey. , .

Bowman, E. H., 1959. The schedule-sequencing problem. *Operations Research*, **7 (5)**, 621–624.

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, **41 (3)**, 157–183.

Brucker, P. and Schlie, R., 1990. Job-shop scheduling with multi-purpose machines. *Computing*, **45 (4)**, 369–375.

Byrne, M. D. and Chutima, P., 1997. Real-time operational control of an fms with full routing flexibility. *International Journal of Production Economics*, **51 (1)**, 109–113.

Castillo, I. and Peters, B. A., 2003. An extended distance-based facility layout problem. *International Journal of Production Research*, **41 (11)**, 2451–2479.

Chan, F. T., Wong, T., and Chan, P., 2004. Equal size lot streaming to job-shop scheduling problem using genetic algorithms. In: Intelligent Control, 2004. Proceedings of the 2004 IEEE International Symposium on. IEEE, pp. 472–476.

Chen, H., Ihlow, J., and Lehmann, C., 1999a. A genetic algorithm for flexible job-shop scheduling. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 2. IEEE, pp. 1120–1125.

Chen, H., Ihlow, J., and Lehmann, C., 1999b. A genetic algorithm for flexible job-shop scheduling. In the proceedings of the 1999 IEEE International Conference on Robotics & Automation. May 1999, Detroit, Michigan, pp. 1120–1125.

Chen, J., Chen, K., Wu, J., and Chen, C., 2007. A study of the flexible job shop scheduling problem with parallel machines and reentrant process. *International Journal of Advanced Manufacturing Technology*, In Press, DOI 10.1007/s00170-007-1227-1.

Chen, M., 2001. A model for integrated production planning in cellular manufacturing systems. *Integrated Manufacturing Systems*, **12 (4)**, 275–284.

Cheng, R., Gen, M., and Tsujimura, Y., 1996. A tutorial survey of job-shop scheduling problems using genetic algorithmsi. representation. *Computers & industrial engineering*, **30 (4)**, 983–997.

Coello, C. A. C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, **191 (11)**, 1245–1287.

Das, S., 1993. A facility layout method for flexible manufacturing systems. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, **31 (2)**, 279–297.

Dauzere-Peres, S. and Lasserre, J.-B., 1997. Lot streaming in job-shop scheduling. *Operations Research*, **45 (4)**, 584–595.

Dauzère-Pérès, S. and Paulli, J., 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, **70**, 281–306.

Defersha, F. and Chen, M., 2008. A linear programming embedded genetic algorithm for an integrated cell formation and lot sizing considering product quality. *European Journal of Operational Research,*, **(187)**, 46–69.

Defersha, F. and Chen, M., 2009a. A simulated annealing algorithm for dynamic system reconfiguration and production planning in cellular manufacturing. *International Journal of Manufacturing Technology and Management,*, **17 (1/2)**, 103–124.

Defersha, F. M. and Chen, M., 2009b. A coarse-grain parallel genetic algorithm for flexible job-shop scheduling with lot streaming. In: Computational Science

and Engineering, 2009. CSE'09. International Conference on. Vol. 1. IEEE, pp. 201–208.

Defersha, F. M. and Chen, M., 2010. A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. *The international journal of advanced manufacturing technology*, **49 (1-4)**, 263–279.

Demir, Y. and İşleyen, S. K., 2013. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*, **37 (3)**, 977–988.

Drira, A., Pierreval, H., and Hajri-Gabouj, S., 2007. Facility layout problems: A survey. *Annual Reviews in Control*, **31 (2)**, 255–267.

Drolet, J., 1989. Scheduling virtual cellular manufacturing systems. Ph.d. thesis, School of Industrial Engineering ,Purdue University, West Lafayette, IN.

Dunker, T., Radons, G., and Westkamper, E., 2005. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *European Journal of Operational Research*, **165 (1)**, 55–69.

Framinan, J. M., Leisten, R., and García, R. R., 2014. Manufacturing Scheduling Systems. Springer,

Ganesan, A., 2007. Modeling of distributed layouts for dynamic period cases. Ph.D. thesis, Wichita State University.

Gao, J., Gen, M., Sun, L., and Zhao, X., 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, **53**, 149–162.

Garey, M. R., Johnson, D. S., and Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, **1 (2)**, 117–129.

Gentry, R. J. and Elms, H., 2009. Firm partial modularity and performance in the electronic manufacturing services industry. *Industry and Innovation*, **16 (6)**, 575–592.

Gindy, N., M.Ratchev, T., and Case, K., 1996. Component grouping for cell formation using resource elements. *International Journal of Production Research*, **34 (3)**, 727–752.

Glover, F. and Kochenberger, G. A., 2003. Handbook of metaheuristics. Springer,

Goetz, W. G. and Egbelu, P. J., 1990. Guide path design and location of load pick-up/drop-off points for an automated guided vehicle system. *International Journal of Production Research*, **28 (5)**, 927–941.

Goldberg, D. E. and Holland, J. H., 1988. Genetic algorithms and machine learning. *Machine learning*, **3 (2)**, 95–99.

Gupta, Y., Gupta, M., Kumar, A., and Sundaram, C., 1996. A genetic algorithm-based approach to cell composition and layout design problems. *International Journal of Production Research*, **34 (2)**, 447–482.

Hamedi, M., Ismailand, N. B., Esmaeilian, G. R., and Ariffin, M., 2012. Developing a method to generate semi-distributed layouts by genetic algorithm. *International Journal of Production Research*, **50 (4)**, 953–975.

Heragu, S. S. and Ekren, B. Y., 2010. Manufacturing facility design and layout. *Wiley Encyclopedia of Operations Research and Management Science*, .

Heragu, S. S. and Kochhar, J. S., 1994. Material handling issues in adaptive manufacturing systems. The Materials Handling Engineering Division 75th Anniversary Commemorative Volume, ASME. New York.

Ho, N. B. and Tay, J. C., 2004. Genace: An efficient cultural algorithm for solving the flexible job-shop problem. In: Evolutionary Computation, 2004. CEC2004. Congress on. Vol. 2. IEEE, pp. 1759–1766.

Ho, N. B., Tay, J. C., and Lai, E. M.-K., 2007. An effective architecture for learning and evolving flexible job-shop schedules. *European Journal of Operational Research*, **179 (2)**, 316–333.

Hosseini Nasab, H., 2014. A hybrid fuzzy-ga algorithm for the integrated machine allocation problem with fuzzy demands. *Applied Soft Computing*, **23**, 417–431.

Hurink, J., Jurisch, B., and Thole, M., 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, **15 (4)**, 205–215.

Irani, S. A. and Huang, H., 2000. Custom design of facility layouts for multiproduct facilities using layout modules. *Robotics and Automation, IEEE Transactions on*, **16 (3)**, 259–267.

Jia, H., Nee, A. Y., Fuh, J. Y., and Zhang, Y., 2003. A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing*, **14 (3-4)**, 351–362.

Johnson, S. M., 1959. Discussion: Sequencing n jobs on two machines with arbitrary time lags. *Management Science*, **5 (3)**, 299–303.

Kacem, I., Hammadi, S., and Borne, P., 2002a. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **32 (1)**, 1–13.

Kacem, I., Hammadi, S., and Borne, P., 2002b. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics*, **32**, 1–3.

Kesen, S. E., Das, S. K., and Güngör, Z., 2010. A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (vmcs). *Computers & Operations Research*, **37 (6)**, 1148–1156.

Kesen, S. E. and Güngör, Z., 2012. Job scheduling in virtual manufacturing cells with lot-streaming strategy: a new mathematical model formulation and a genetic algorithm approach. *Journal of the Operational Research Society*, **63 (5)**, 683–695.

Khaewsukkho, S., 2008. New approaches for design of high-mix low-volume facilities. Ph.D. thesis, The Ohio State University.

Kirkpatrick, S., Gelatt, C., and Vecchi, M., 1983. Optimization by simulated annealing. *Science*, **220**, 671–680.

Kochhar, J. and Heragu, S., 1999. Facility layout design in a changing environment. *International Journal of Production Research*, **37 (11)**, 2429–2446.

Koopmans, T. C. and Beckmann, M., 1957. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, , 53–76.

Koski, J., 1981. Multicriterion optimization in structural design. Tech. rep., DTIC Document.

Koski, J. and Silvennoinen, R., 1987. Norm methods and partial weighting in multicriterion optimization of structures. *International Journal for Numerical Methods in Engineering*, **24 (6)**, 1101–1121.

Krishna, K., Vignesh, K., and Jithavech, I., 2009. Simulation modeling and analysis of distributed and process process layouts. *California Journal of Operations Management*, **7 (1)**, 65–56.

Kusiak, A. and Heragu, S. S., 1987. The facility layout problem. *European Journal of operational research*, **29 (3)**, 229–251.

Lahmer, M. and Benjaafar, S., 2005. Design of distributed layouts. *IIE Transactions*, **37 (4)**, 303–318.

Langston, M. A., 1987. Interstage transportation planning in the deterministic flow-shop environment. *Operations Research*, **35 (4)**, 556–564.

Lee, K.-M., Yamakawa, T., and Lee, K.-M., 1998. A genetic algorithm for general machine scheduling problems. In: Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on. Vol. 2. IEEE, pp. 60–66.

Lee, S.-Y. and Lee, K., 1996. Synchronous and asynchronous parallel simulated annealing with multiple markov chains. *IEEE Trans Parallel Distrib Sys*, **7**, 903–1007.

Liu, H., Abraham, A., and Grosan, C., 2007. A novel variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems. In: Digital Information Management, 2007. ICDIM'07. 2nd International Conference on. Vol. 1. IEEE, pp. 138–145.

Mak, K.-L., Peng, P., Wang, X., and Lau, T., 2007. An ant colony optimization algorithm for scheduling virtual cellular manufacturing systems. *International Journal of Computer Integrated Manufacturing*, **20 (6)**, 524–537.

Manne, A. S., 1960. On the job-shop scheduling problem. *Operations Research*, **8 (2)**, 219–223.

Marcotte, S., Montreuil, B., and Lefrancois, P., 1998. Holographic layout design: Towards manufacturing agility. 7th Annual Industrial Engineering Research Conference. Banff,Alberta,Canada.

Marler, R. T. and Arora, J. S., 2010. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, **41 (6)**, 853–862.

Marler, T., 2009a. A study of multi-objective optimization methods: for engineering applications. VDM Publishing, pp. 30–43.

Marler, T., 2009b. A study of multi-objective optimization methods: for engineering applications. VDM Publishing, pp. 45–48.

Meller, R. D. and Gau, K.-Y., 1996. The facility layout problem: recent and emerging trends and perspectives. *Journal of manufacturing systems*, **15 (5)**, 351–366.

Meller, R. D., Narayanan, V., and Vance, P. H., 1998. Optimal facility layout design. *Operations Research Letters*, **23 (3)**, 117–127.

Meng, G., Heragu, S., and Zijm, H., 2004. Reconfigurable layout problem. *International Journal of Production Research*, **42 (22)**, 4709–4729.

Mesghouni, K., Hammadi, S., and Borne, P., 1997. Evolution programs for job-shop scheduling. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. Vol. 1. IEEE, pp. 720–725.

Mitten, L., 1959. Sequencing n jobs on two machines with arbitrary time lags. *Management science*, **5 (3)**, 293–298.

Montreuil, B., 1999. Fractal layout organization for job shop environments. *International journal of production research*, **37 (3)**, 501–521.

Montreuil, B., LeFrancois, P., Marcotte, S., and Venkatadri, U., 1993. Holographic layout of manufacturing systems operating in chaotic environments. Tech. rep., Technical Report.

Montreuil, B. and Venkatadri, U., 1991. Strategic interpolative design of dynamic manufacturing systems layout. *Management Science*, **37 (6)**, 682–694.

Montreuil, B., Venkatadri, U., and Lefrançois, P., 1991. Holographic layout of manufacturing systems. , .

Nageshwaraniyer, S., Khilwani, N., Tiwari, M., and Shankar, D., R.and Ben-Arieh, 2013. Solving the design of distributed layout problem using forecast windows: A hybrid algorithm approach. *Robotics and Computer-Integrated Manufacturing*, **29 (1)**, 128138.

Palekar, U. S., Batta, R., Bosch, R. M., and Elhence, S., 1992. Modeling uncertainties in plant layout problems. *European Journal of Operational Research*, **63 (2)**, 347–359.

Panwalkar, S. S., Dudek, R. A., and Smith, M. L., 1973. Sequencing research and the industrial scheduling problem. In S. E. Elmaghraby (Ed.), Symposium on the theory of scheduling and its applications. Springer-Verlag, p. 29.

Paredis, J., 1992. Exploiting constraints as background knowledge for genetic algorithms: A case-study for scheduling. In: PPSN. pp. 231–240.

Park, Y.-T. and Wemmerlöv, U., 1995. A comparative study of cell formation techniques: Experimental findings on data-dependency and solution quality. *Decision Sciences*, **26 (4)**, 475–502.

Paulli, J., 1995. A hierarchical approach for the fms scheduling problem. *European Journal of Operational Research*, **86 (1)**, 32–42.

Pezzella, F., Morganti, G., and Ciaschetti, G., 2008a. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, **35 (10)**, 3202–3212.

Pezzella, F., Morganti, G., and Ciaschetti, G., 2008b. A genetic algorithm for the flexible job-shop scheduling problem. *Computers and Operations Research*, **35**, 3202–3212.

Potts, C. N. and Van Wassenhove, L. N., 1992. Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. *Journal of the Operational Research Society*, , 395–406.

Puchinger, J. and Raidl, G. R., 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. Springer,

Rao, S. and Freiheit, T., 1991. A modified game theory approach to multiobjective optimization. *Journal of Mechanical Design*, **113 (3)**, 286–291.

Rosenblatt, M. J. and Kropp, D. H., 1992. The single period stochastic plant layout problem. *IIE transactions*, **24 (2)**, 169–176.

Ruiz, R., Şerifoglub, F. S., and Urlings, T., 2008. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, **35**, 1151–1175.

Sabuncuoglu, I. and Hommertzheim, D. L., 1992. Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system. *The International Journal of Production Research*, **30 (5)**, 1059–1079.

Safaei, N. and Tavakkoli-Moghaddam, R., 2009. Integrated multi-period cell formation and subcontracting production planning in dynamic cellular manufacturing systems. *International Journal of Production Economics*, **120 (2)**, 301–314.

Sahni, S. and Gonzalez, T., 1976. P-complete approximation problems. *Journal of the ACM (JACM)*, **23 (3)**, 555–565.

Saidi, M. and Fattahi, P., 2007. Flexible job shop scheduling with tabu search algorithm. *International Journal of Advanced Manufacturing Technology*, **35**, 563–570.

Shafigh, F., Defersha, F. M., and Moussa, S. E., 2015. A mathematical model for the design of distributed layout by considering production planning and system reconfiguration over multiple time periods. *Journal of Industrial Engineering International*, , 1–13.

Shahbazi, H., Ghahyazi, A. E., and Zeinali, F., 2013. Quadratic assignment problem. *Journal of American Science*, **9 (8)**, 2013.

Sivanandam, S. and Deepa, S., 2007. Introduction to genetic algorithms. Springer,

Sivrikaya-Şerifoğlu, F. and Ulusoy, G., 1998. A bicriteria two-machine permutation flowshop problem. *European journal of operational research*, **107 (2)**, 414–430.

Solimanpur, M. and Jafari, A., 2008. Optimal solution for the two-dimensional facility layout problem using a branch-and-bound algorithm. *Computers & Industrial Engineering*, **55 (3)**, 606–619.

Sule, D. R., 1994. Manufacturing facilities: location, planning, and design. PWS, Boston,

Sundhararajan, S. and Pahwa, A., 1994. Optimal selection of capacitors for radial distribution systems using a genetic algorithm. *Power Systems, IEEE Transactions on*, **9 (3)**, 1499–1507.

Taghavi, A. and Murat, A., 2011. A heuristic procedure for the integrated facility layout design and flow assignment problem. *Computers & Industrial Engineering*, **61 (1)**, 55–63.

Teghem, J., Pirlot, M., and Antoniadis, C., 1995. Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem. *Journal of Computational and Applied Mathematics*, **64**, 91–102.

Tompkins, J. A., White, J. A., Bozer, Y. A., Frazelle, E. H., and Tanchoco, J. M. A., 1996. Facilities Planning. Wiley, New York, p. 6.

Tuncel, G., 2012. An integrated modeling approach for shop-floor scheduling and control problem of flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, **59 (9-12)**, 1127–1142.

Urban, T. L., Chiang, W.-C., and Russell, R. A., 2000. The integrated machine allocation and layout problem. *International journal of production research*, **38 (13)**, 2911–2930.

Vaessens, R. J. M., Aarts, E. H., and Lenstra, J. K., 1996. Job shop scheduling by local search. *INFORMS Journal on Computing*, **8 (3)**, 302–317.

Venkatadri, a. R. R. L. a. M. B., Uday, 1997. A design methodology for fractal layout organization a design methodology for fractal layout organization. *lIE Transactions*, **29 (10)**, 911–924.

Wagner, H. M., 1959. An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly*, **6 (2)**, 131–140.

Xia, W. and Wu, Z., 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, **48**, 409–425.

Xing, L., Chen, Y., and Yang, K., 2008. Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling. *Applied Soft Computing*, **In Press (doi:10.1016/j.asoc.2008.04.013)**.

Zadeh, L., 1963. Optimality and non-scalar-valued performance criteria. *Automatic Control, IEEE Transactions on*, **8 (1)**, 59–60.

Zhang, G., Gao, L., and Shi, Y., 2011. An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, **38 (4)**, 3563–3573.

Zhang, H. and Gu, M., 2008. Modeling job shop scheduling with batches and setup times by timed petri nets. *Mathematical and Computer Modelling*, **In Press (doi:10.1016/j.mcm.2008.03.010)**.

Appendix A

Input Data for Problem-1

Table A.1: Resource Elements data

Resource Element r	Indices of machines having resource element r			
	Case 1	Case 2	Case 3	Case 4
1	(1, 2, 3, 4, 5, 6)	(1, 2, 4, 6)	(1, 6)	(1)
2	(1, 2, 3, 4, 5, 6)	(1, 3, 5)	(2, 5)	(2)
3	(1, 2, 3, 4, 5, 6)	(2, 4, 6)	(3)	(3)
4	(1, 2, 3, 4, 5, 6)	(1, 3, 5, 6)	(4)	(4)
5	(7, 8, 9, 10)	(7, 8, 10)	(8)	(5)
6	(7, 8, 9, 10)	(9, 10)	(7, 10)	(6)
7	(7, 8, 9, 10)	(7, 8, 10)	(9)	(7)
8	(7, 8, 9, 10)	(8, 9)	(7, 10)	(8)
9	(11, 12, 13, 14, 15, 16)	(11, 14, 15)	(14, 15)	(9)
10	(11, 12, 13, 14, 15, 16)	(11, 13, 16)	(11, 16)	(10)
11	(11, 12, 13, 14, 15, 16)	(12, 14, 16)	(16)	(11)
12	(11, 12, 13, 14, 15, 16)	(11, 13, 15)	(12, 13)	(12)
13	(11, 12, 13, 14, 15, 16)	(11, 12, 14)	(11, 15, 16)	(13)
14	(17, 18, 19)	(18, 19)	(18)	(14)
15	(17, 18, 19)	(17, 19)	(17)	(15)
16	(17, 18, 19)	(18, 19)	(19)	(16)
17	(20, 21, 22)	(20, 22)	(22)	(17)
18	(20, 21, 22)	(21, 22)	(20)	(18, 21)
19	(20, 21, 22)	(20, 21)	(21)	(19, 22)
20	(20, 21, 22)	(20, 22)	(20, 22)	(20)

Table A.2: Processing data for the parts

								Operation data (r, U_p)				
Part	Θ_p	$\hat{\Theta}_p$	H_p	F_p	S_p	N_p	O_p	o = 1	o = 2	o = 3	o = 4	o = 5
1	6	12	5	2	300	2	2	(14, 1)	(8, 3)			
2	10	40	5	3	200	2	3	(17, 2)	(19, 1)	(1, 2)		
3	6	18	2	2	150	2	4	(4, 2)	(12, 3)	(4, 2)	(2, 3)	
4	4	16	4	1	200	2	2	(14, 1)	(18, 2)			
5	6	18	4	2	300	2	2	(6, 2)	(14, 1)			
6	8	24	3	2	150	2	2	(13, 3)	(14, 2)			
7	8	24	3	2	350	2	4	(3, 2)	(1, 1)	(16, 3)	(2, 1)	
8	10	30	3	1	250	2	5	(0, 2)	(9, 1)	(13, 3)	(1, 2)	(4, 2)
9	4	16	5	3	400	2	5	(6, 2)	(4, 2)	(10, 2)	(3, 2)	(18, 2)
10	6	18	3	1	350	2	4	(18, 2)	(6, 1)	(19, 1)	(19, 2)	
11	2	6	2	3	150	2	3	(12, 1)	(8, 1)	(1, 3)		
12	4	12	5	2	350	2	3	(6, 3)	(15, 1)	(4, 2)		
13	4	12	4	3	250	2	3	(11, 2)	(9, 2)	(12, 2)		
14	2	6	4	2	350	2	4	(16, 3)	(14, 2)	(7, 2)	(14, 1)	
15	4	8	3	3	250	2	4	(17, 2)	(4, 1)	(13, 2)	(7, 2)	
16	10	40	3	2	200	2	4	(2, 2)	(12, 2)	(13, 2)	(3, 3)	
17	2	6	4	3	200	2	3	(16, 2)	(12, 3)	(1, 2)		
18	10	30	3	1	250	2	2	(19, 2)	(11, 2)			
19	8	16	6	2	350	2	5	(3, 2)	(17, 1)	(14, 3)	(6, 1)	(1, 1)
20	4	16	4	2	200	2	3	(16, 2)	(9, 2)	(6, 2)		
21	6	18	4	3	300	2	4	(10, 2)	(6, 1)	(6, 2)	(18, 3)	
22	6	12	3	1	350	2	4	(14, 3)	(18, 2)	(1, 1)	(10, 3)	
23	8	24	2	3	400	2	4	(18, 2)	(17, 1)	(6, 2)	(10, 1)	
24	4	12	3	2	350	2	4	(6, 1)	(10, 3)	(2, 2)	(7, 2)	
25	4	16	3	2	350	2	3	(15, 2)	(7, 2)	(10, 2)		

Note: Operation data (r, U_p) is the index of the required resource element r and unit processing time U_p for the corresponding operation.

Table A.3: Demand data for the parts

	Demand $D_{p,t}$			
Part	$t = 1$	$t = 2$	$t = 3$	$t = 4$
1	50	100	0	650
2	0	50	550	200
3	150	300	300	0
4	400	0	150	350
5	0	100	450	450
6	250	600	0	0
7	550	0	0	200
8	0	100	400	100
9	650	150	700	100
10	0	350	0	0
11	550	250	0	350
12	450	0	0	0
13	0	450	200	50
14	100	650	600	0
15	400	150	0	0
16	0	100	700	250
17	750	0	300	200
18	200	700	700	0
19	0	0	200	0
20	150	0	100	200
21	150	0	0	500
22	700	700	150	450
23	700	450	250	300
24	600	100	450	200
25	500	450	350	0

Table A.4: Machine relocation cost per unit distance

m	G_m	m	G_m	m	G_m	m	G_m
1	80	7	100	13	80	18	80
2	80	8	80	14	80	19	60
3	60	9	80	15	100	20	80
4	80	10	80	16	80	21	100
5	60	11	80	17	60	22	80
6	80	12	100				

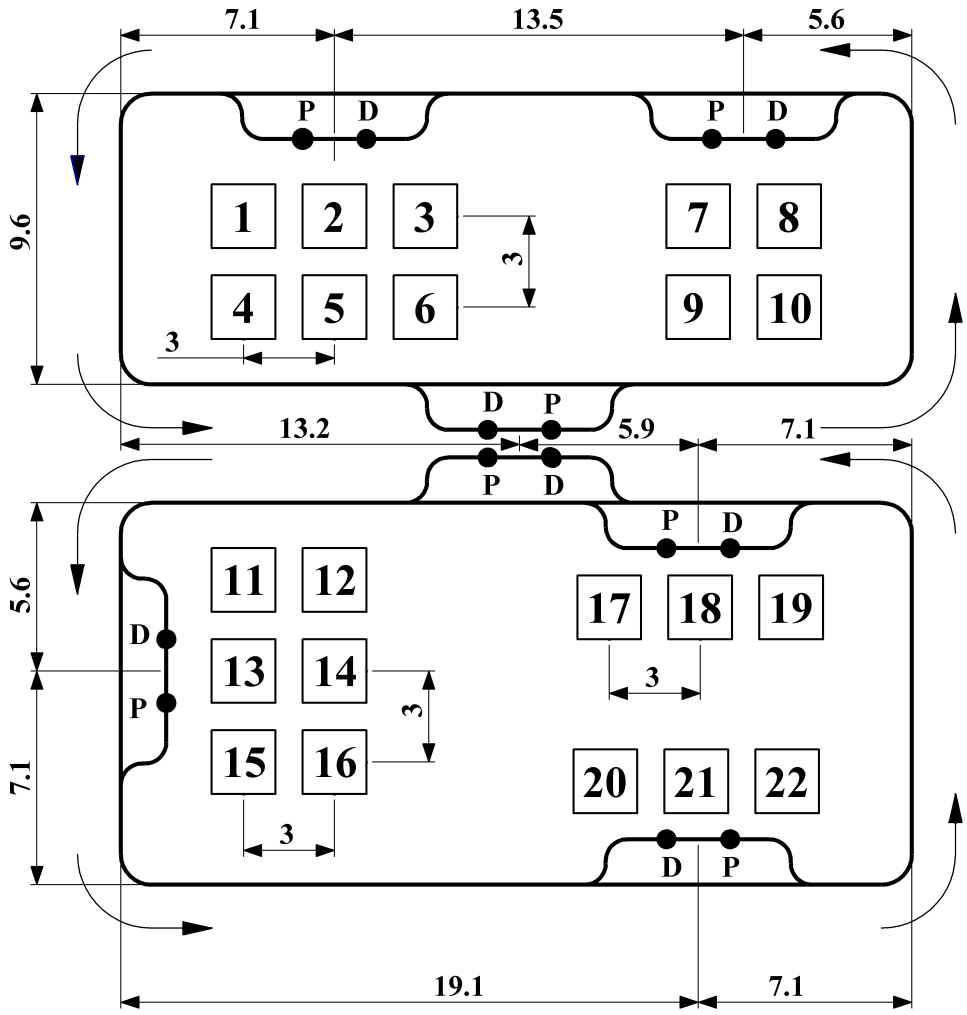


Figure A.1: Layout showing AGV path and locations for machines - dimensions are in unit distance

Table A.5: Machine location for the functional and five arbitrary generated distributed layouts

Machine index m at location $l = 1$ to 22					
Functional	Distributed Layouts				
Layout $l = m$	DL1	DL2	DL3	DL4	DL5
1	10	4	19	12	22
2	4	5	16	15	11
3	20	12	2	4	5
4	2	3	14	8	10
5	5	13	18	17	1
6	18	8	15	20	18
7	7	20	1	1	2
8	19	17	13	19	21
9	22	6	21	3	13
10	13	11	7	14	20
11	15	9	12	22	15
12	16	21	3	5	6
13	6	7	11	7	8
14	11	19	4	2	7
15	3	1	17	18	14
16	8	16	5	16	19
17	21	10	6	11	4
18	14	22	20	9	3
19	17	2	9	10	12
20	12	15	10	6	9
21	1	18	8	21	16
22	9	14	22	13	17

Table A.6: Material handling distance between locations l and l', $E_{l,l'}$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	3	6	3	6	9	61	61	64	64	55	58	52	55	55	58	108	105	108	81	78	81
2	3	0	3	6	3	6	58	58	61	61	52	55	48	52	52	55	105	102	105	78	75	78
3	6	2	0	9	6	3	61	61	64	64	55	58	52	55	55	58	108	105	108	81	78	81
4	3	6	9	0	3	6	64	64	67	67	58	61	55	58	58	61	111	108	111	84	81	84
5	6	3	6	3	0	3	61	61	64	64	55	58	52	55	55	58	108	105	108	81	78	81
6	9	6	3	6	3	0	64	64	67	67	58	61	55	58	58	61	111	108	111	84	81	84
7	17	14	17	20	17	20	0	3	3	6	65	68	62	65	65	68	118	115	118	91	88	91
8	17	14	17	20	17	20	3	0	6	3	65	68	62	65	65	68	118	115	118	91	88	91
9	20	17	20	23	20	23	3	6	0	3	68	71	65	68	68	71	121	118	121	94	91	94
10	20	17	20	23	20	23	6	3	3	0	68	71	65	68	68	71	121	118	121	94	91	94
11	107	104	107	110	107	110	90	90	93	93	0	3	3	6	6	9	59	56	59	32	29	32
12	110	107	110	113	110	113	93	93	96	96	3	0	6	3	9	6	62	59	62	35	32	35
13	104	101	104	107	104	107	87	87	90	90	3	6	0	3	3	6	56	53	56	29	26	29
14	107	104	107	110	107	110	90	90	93	93	6	3	3	0	6	3	59	56	59	32	29	32
15	107	104	107	110	107	110	90	90	93	93	6	9	3	6	0	3	59	56	59	32	29	32
16	110	107	110	113	110	113	93	93	96	96	9	6	6	3	3	0	62	59	62	35	32	35
17	54	51	54	57	54	57	37	37	40	40	31	34	28	31	31	34	0	3	6	57	54	57
18	51	48	51	54	51	54	34	34	37	37	28	31	25	28	28	31	3	0	3	54	51	54
19	54	51	54	57	54	57	37	37	40	40	31	34	28	31	31	34	6	3	0	57	54	57
20	81	78	81	84	81	84	64	64	67	67	58	61	55	58	58	61	33	30	33	0	3	6
21	78	75	78	81	78	81	61	61	64	64	55	58	52	55	55	58	30	27	30	3	0	3
22	81	78	81	84	81	84	64	64	67	67	58	61	55	58	58	61	33	30	33	6	3	0

Table A.7: Machine relocation distance between location l and l', $E'_{l,l'}$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	3	6	3	4	7	15	18	15	18	12	12	15	15	18	18	18	20	22	22	24	25
2	3	0	3	4	3	4	12	15	12	15	12	12	15	15	18	18	16	18	20	21	22	24
3	6	3	0	7	4	3	9	12	10	12	13	12	16	15	19	18	14	16	18	20	21	22
4	3	4	7	0	3	6	15	18	15	18	9	10	12	12	15	15	16	18	21	20	22	24
5	4	3	4	3	0	3	12	15	12	15	10	9	10	12	15	15	13	16	18	18	20	22
6	7	4	3	6	3	0	10	12	9	12	11	10	13	12	16	15	12	13	16	17	18	20
7	15	12	9	15	12	10	0	3	3	4	19	17	21	19	24	22	13	13	13	19	19	19
8	18	15	12	18	15	12	3	0	4	3	22	19	24	21	26	24	14	13	13	13	19	19
9	15	12	10	15	12	9	3	4	0	3	18	15	19	17	21	19	10	10	10	16	16	16
10	18	15	12	18	15	12	4	3	3	0	20	18	22	19	24	22	12	10	10	17	16	16
11	12	12	13	9	10	11	19	22	18	20	0	3	3	4	6	7	12	15	18	14	16	19
12	12	12	12	10	9	10	17	19	15	18	3	0	4	3	7	6	9	12	15	11	14	16
13	15	15	16	12	10	13	21	24	19	22	3	4	0	3	3	4	12	15	18	13	15	18
14	15	15	15	12	12	12	19	21	17	19	4	3	3	0	4	3	9	12	15	10	13	15
15	18	18	19	15	15	16	24	26	21	24	6	7	3	4	0	3	13	16	19	12	15	18
16	18	18	18	15	15	15	22	24	19	22	7	6	4	3	3	0	10	13	16	9	12	15
17	18	16	14	16	13	12	13	14	10	12	12	9	12	9	13	10	0	3	6	6	6	8
18	20	18	16	18	16	13	13	13	10	10	15	12	15	12	16	13	3	0	3	6	6	6
19	22	20	18	21	18	16	13	13	10	10	18	15	18	15	19	16	6	3	0	8	6	6
20	22	21	20	20	18	17	19	13	16	17	14	11	13	10	12	9	6	6	8	0	3	6
21	24	22	21	22	20	18	19	19	16	16	16	14	15	13	15	12	6	6	6	3	0	3
22	26	24	22	24	22	20	19	19	16	16	19	16	18	15	18	15	8	6	6	6	3	0