Journal of the Korean Institute of Industrial Engineers
Vol. 35, No. 1, pp. 15-39, March 2009.

Survey of Evolutionary Algorithms in Advanced Planning and Scheduling

Mitsuo Gen[1†] · Wenqiang Zhang[1] · Lin Lin[2]

[1]Graduated School of Information, Production and Systems, Waseda University, 808-0135, Japan

[2]Information, Production and Systems Research Center, Waseda University, 808-0135, Japan

Advanced planning and scheduling (APS) refers to a manufacturing management process by which raw materials and production capacity are optimally allocated to meet demand. APS is especially well-suited to environments where simpler planning methods cannot adequately address complex trade-offs between competing priorities. However, most scheduling problems of APS in the real world face both inevitable constraints such as due date, capability, transportation cost, set up cost and available resources.
In this survey paper, we address three crucial issues in APS, including basic scheduling model, job-shop scheduling (JSP), assembly line balancing (ALB) model, and integrated scheduling models for manufacturing and logistics. Several evolutionary algorithms which adapt to the problems are surveyed and proposed; some test instances based on the practical problems demonstrate the effectiveness and efficiency of evolutionary approaches.

Keywords: Advanced Planning and Scheduling (APS), Evolutionary Algorithm, Job-Shop Scheduling (JSP), Assembly Line Balancing (ALB), Integrated Scheduling Models for Manufacturing and Logistics.

1. Introduction

The global concepts of manufacturing systems throughout modern history were in close connection with the then principles and findings of the science, philosophy, and arts. It can be noticed that the manufacturing concepts reflected the principles, criteria, and values generally accepted by the society as the most important. For example, in the eighteenth, nineteenth, and the first half of the twentieth century, the scientific facts mainly exposed the unchanged ability, determinism, rationality, exactness, and causality. That period can be characterized as the era of predominance of production. In the second half of the twentieth century, the advanced information technology assured the formal conditions for the expansion of various organizational forms. The second half of the twentieth century can thus be characterized as the era when organizational aspects were prevailing.

Future manufacturing concepts will have to be adapted to the needs of the modern society and particularly, to the ecosystem more than ever. Unfortunately, the term adaptability is still bereft today of an important component; the adaptability still has a particularly mechanistic-technological connotation, whereas the component considering the adaptability to the ecosystem is almost completely neglected.

For this reason, in the existing manufacturing systems for optimization of production of goods, particularly the technological parameters are considered which are measurable or which we want to measure, whereas the global interaction of the goods with the ecosystem is much less considered. Therefore, today it is possible to speak particularly of the local mechanistic-technological optimum of the goods, whereas the global optimum, affected by several influencing parameters and mutual relations, is not reached in most cases. In present manufacturing systems, particularly the deterministic approaches are used for synchronization of material, energy, and information flows. It means that the methods based on the exact mathe-

This work is partly supported by the Ministry of Education, Science and Culture, the Japanese Government : Grant-in-Aid for Scientific Research (No.19700071, No.20500143, A08210500).

† Corresponding author : Mitsuo Gen, Graduated School of Information, Production and Systems, Waseda University, 808-0135, Japan, E-mail : gen@waseda.jp

Received August 1, 2008; Revision Received February 1, 2009; Accepted February 3, 2009.

matical findings and the rules of logic are used for modeling, optimization, and functioning of systems. However, production is a very dynamic process with many unexpected events and continuously emerging new requirements. Therefore, exact description of a system is often not possible by conventional methods. Mathematical models are often derived by making simplifying assumptions. As a consequence, the model of the system is not in accordance with functioning of the real system. Such models of the systems are frequently not suitable and flexible enough to respond efficiently to fast changes in the environment.

The existing manufacturing concepts cannot successfully respond to the abovementioned problems to which modern production and society as a whole will be exposed in the future more than ever. However, in many different areas of science and technology it has been possible recently to notice the shift towards the conceiving of integrated systems capable of learning and efficiently responding to increasing complexity, unpredictability, and changeability of the environment. During the learning process, the system behavior gradually improves. Machine learning as area of artificial intelligence is increasingly gaining importance. Several successful integrated systems have been conceived by the methods of machine learning. Recently, some interesting manufacturing concepts and approaches to problem solving based on learning, self-organization, and on the bottom-up organizational principle were also proposed. The intelligence, order and consequent, efficiency of those systems develop step by step and emerge due to interactions of the basic components (entities) of the systems with the environment. However, more research will be required for the concepts to take roots in real environments to the full extent.

In an *integrated manufacturing system* (IMS), the functional areas involved in providing the finished products to customers are integrated in a single system. These areas vary from manufacture product to distribution and from product design to facility management as shown in <Figure 1>. To find the optimal solutions in those fields gives rise to complex combinatorial optimization problems; unfortunately, most of them fall into the class of NP-hard problems. Hence to find a satisfactory solution in an acceptable time span is crucial for the performance of IMS. Genetic algorithms have turned out to be potent methods to solve such kinds of optimization problems. In an IMS the basic problems indicated in <Figure 1> are likely to use evolutionary techniques.

1. *Design* : Design problems generally have to be decided only once in IMS, and they form some kinds of fixed inputs for other subsequent problems, such as manufacturing and planning problems. Typically, design problems in an integrated manufacturing system include layout design, assembly planning, group technology and so on.
2. *Planning* : Compared to scheduling problems, planning problems have a longer horizon. Hence the demand information needed to find the optimal solution for a planning problem comes from forecasting rather than arrived orders. Process planning, operation sequencing, production planning and assembly line balancing fall into the class of planning problems.
3. *Manufacturing* : In manufacturing, there are two kind of essential issues : scheduling and routing. Machining, assembly, material handling and other manufacturing functions are performed to the best efficiency. Such kinds of problems are generally triggered by a new order.
4. *Distribution* : The efficient distribution of products is very important in IMS, as transportation costs become a nonnegligible part of the purchase price of products in competitive markets. This efficiency is achieved through sophisticated logistic network design and efficient traffic routing.

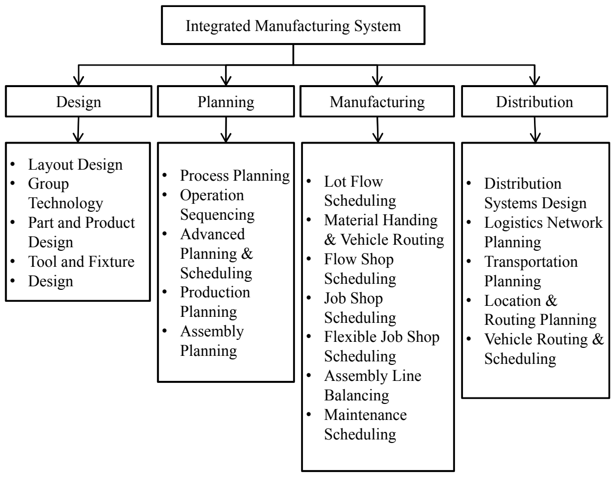
Integrated Manufacturing System
Design
Planning
Manufacturing
Distribution
• Layout Design
• Group Technology
• Part and Product Design
• Tool and Fixture
• Design
• Process Planning
• Operation Sequencing
• Advanced Planning & Scheduling
• Production Planning
• Assembly Planning
• Lot Flow Scheduling
• Material Handing & Vehicle Routing
• Flow Shop Scheduling
• Job Shop Scheduling
• Flexible Job Shop Scheduling
• Assembly Line Balancing
• Maintenance Scheduling
• Distribution Systems Design
• Logistics Network Planning
• Transportation Planning
• Location & Routing Planning
• Vehicle Routing & Scheduling

Figure 1. Basic problems in an integrated manufacturing system

Advanced planning and scheduling (APS) includes a range of capabilities from finite-capacity scheduling at the shop floor level through to constraint-based planning in IMS. APS is a new revolutionary step in enterprise and inter-enterprise planning. It is revolutionary, due to the technology and because APS utilises planning and scheduling techniques that consider a wide range of constraints to produce an optimized plan (Eck, 2003) :

- Material availability
- Machine and labour capacity
- Customer service level requirements (due dates)
- Inventory safety stock levels
- Cost
- Distribution requirements
- Sequencing for set-up efficiency

Basing on the above description, considering characteristic of different structure of problems and diversity of solutions to problems, three topics, *job-shop scheduling problem* (JSP) model, *assembly line balancing* (ALB) model and integrated scheduling models for manufacturing and logistics, are selected in this study. JSP is a basic scheduling model in APS, which concerns to determinate the operation sequences and to minimize the makespan. The ALB not only include the operation sequence between stations, but also need to consider that how to group the operations among stations so that the precedence relations are not violated and a given objective function is optimized. The approaches to ALB models are also suitable for other scheduling problem. The integrated scheduling models for manufacturing and logistics not only consider the process scheduling system with manufacturing constraints, but also concern the transportation scheduling problem between plants, customers and so on.

The rest of the survey paper is organized as follows : in Section 2, we introduce JSP model, and give conventional heuristics, and genetic representations for JSP, that are useful for design evolutionary algorithms to APS. In Section 3, we give a survey of assembly line balancing model, and propose a hybrid genetic approach for assembly line balancing model. In Section 4, we introduce an advanced APS model, integrated scheduling model for manufacturing and logistics. This paper give the conclusion follows in Section 5.

2. Job-Shop Scheduling Model

Following Blazewicz *et al.* (1994), scheduling problems can be broadly defined as "the problems of the allocation of resources over time to perform a set of tasks". The scheduling literature is full of very diverse scheduling problems (Brucker, 1998, French, 1982). The JSP concerns determination of the operation sequences on the machines so that the makespan is minimized. It consists of several assumptions as follows (Cheng *et al.,* 1996) :

A1 : Each machine processes only one job at a time.
A2 : Each job is processed on one machine at a time.
A3 : A job does not visit the same machine twice.
A4 : The processing time of each operation has been determined.
A5 : There are no precedence constraints among operations of different jobs.
A6 : Operations cannot be interrupted.
A7 : Neither release times nor due dates are specified.

This problem has already been confirmed as one of the NP-hard problems. There are n jobs and m machines to be scheduled; furthermore each job is composed of a set of operations and the operation order on machines is prespecified, and each operation is characterized by the required machine and the fixed processing time (Cheng *et al.*, 1996, Jain and Meeran, 1998, Gen and Cheng, 2002, Gao *et al.*, 2007, 2008).

Opinion in this field is mixed about who first proposed JSP in its current form. Roy and Sussmann were the first to propose the disjunctive graph representation (Roy and Sussmann, 1964), and Balas was the first to apply an enumerative approach based on the disjunctive graph. Since then many researchers have tried various strategies for solving this problem (Balas, 1969). Dispatching rules was adapted in Grabot's work (Grabot and Geneste, 1994), and Yang also used neural networks combined with a heuristics approach to solve the problem (Yang and Wang, 2000); moreover some researchers especially Gen and Cheng using hybrid GA also obtained good solutions (Gen and Cheng, 2002, Cheng *et al.*, 1999).

Recently, many researchers tried to adapt different meta-heuristic approaches to obtain a near-optimal solution of JSP. Monch and Driessel considered distributed versions of a modified shifting bottleneck heuristic to solve complex job shops. Nowicki and Smutnicki provide a new approximate Tabu Search (TS) algorithm that is based on the big valley phenomenon, and uses some elements of so-called path relinking technique as well as new theoretical properties of neighborhoods (Nowicki and Smutnicki, 2005). Tavakkoli-Moghaddam *et al.* (2005) used *neural network* (NN) approach to generate initial feasible solutions and adapted a simulated annealing (SA) algorithm to improve the quality and performance of the solution.

To improve the efficiency for finding better solutions in searching space, some special technical local searches have been adapted in JSP. Ida and Osawa (2005) reformed the traditional left shift to short the idle time, and formulated an algorithm called Eshift. Gonçalves *et al.* (2005) proposed another technique based on the critical path to confirm the search space, and swapped the operations in critical block, and this approach can also improve the efficiency of algorithm for finding active schedule.

2.1 Mathematical Formulation of JSP

Notation

Indices

i, l : index of job, $i, l = 1, 2, \cdots, n$

j, h : index of machine, $j, h = 1, 2, \cdots, m$

k : index of operation, $k = 1, 2, \cdots, m$

Parameters

n : total number of jobs

m : total number of machines

t_M : makespan

M_j : the j-th machine

J_i : the i-th job, $i = 1, 2, \cdots, n$

o_{ikj} : the k-th operation of job J_i operated on machine M_j

p_{ikj} : processing time of operation o_{ikj}

Decision Variables

t_{ikj} : completion time of operation o_{ikj} on machine M_j for each job J_i

The JSP we are treating is to minimize the makespan, so the problem could be described as an n-job m-machine JSP by simple equations as follows :

$$\min \ t_M = \max_{ikj}\{t_{ikj}\} \tag{1}$$

$$\text{s.t.} \ \ t_{i,k-1,h} + p_{ikj} \le t_{ikj}, \qquad \forall i,\ k,\ h,\ j \tag{2}$$

$$t_{ikj} \ge 0, \qquad \forall i,\ k,\ j \tag{3}$$

The objective function at Eq. 1 is to minimize the makespan. The constraint at Eq. 2 is the operation precedence constraint, the k-1-th operation of job i should be processed before the k-th operation of the same job.

2.2 Conventional Heuristics for JSP

Job-shop scheduling is one of the hardest combinatorial optimization problems. Since job-shop scheduling is a very important everyday practical problem, it is therefore natural to look for approximation methods that produce an acceptable schedule in useful time. The heuristic procedures for a job-shop problem can be roughly classified into two classes :

- One-pass heuristic
- Multi-pass heuristic

One-pass heuristic simply builds up a single complete solution by fixing one operation in schedule at a time based on *priority dispatching rules*. There are many rules for choosing an operation from a specified subset to be scheduled next. This heuristic is fast, and usually finds solutions that are not too bad. In addition, one-pass heuristic may be used repeatedly to build more sophisticated multi-pass heuristic in order to obtain better schedules at some extra computational cost (Gen *et al.*, 2008). <Table 1> consists of some of the priority rules commonly used in practice.

Table 1. A list of job shop dispatch rules

Rule	Description
SPT (Shortest Processing Time)	Select the operation with the shortest processing time
LPT (Longest Processing Time)	Select an operation with longest processing time
LRT (Longest Remaining Processing Time)	Select the operation belonging to the job with the longest remaining processing time
SRT (Shortest Remaining Processing Time)	Select the operation belonging to the job with the shortest remaining processing time
LRM (LRT excluding the operation under consideration)	Select the operation belonging to the job with the longest remaining processing time excluding the operation under consideration

While one-pass heuristics limit themselves to constructing a single solution, multipass heuristics (also called search heuristics) try to get much better solutions by generating many of them, usually at the expense of a much higher computation time. Techniques like branch-and-bound method and dynamic programming can guarantee an optimal solution, but are not practical for large-scale problems.

Randomized heuristics are an early attempt to provide more accurate solutions (Baker, 1974). The idea of randomized dispatch is to start with a family of dispatching rules. At each selection of an operation to run, choose the dispatching rule randomly, repeated throughout an entire schedule generation. Repeat the entire process several times and choose the best result. Various researchers tried to improve on the randomization approach. One change is to have a learning process so that more successful dispatching rules will have higher chances of being selected in the future. Morton and Pentico proposed a guided random approach. The *guided* means that an excellent heuristic is needed first, to "explore" the problem and provide good guidance as to where to search.

The *shifting bottleneck heuristic* from Adams *et al.* (1987) is probably the most powerful procedure known up to now among all heuristics for the job-shop scheduling problem. It sequences the machines one by one, successively, taking each time the machine identified as a bottleneck among the machines not yet sequenced. Every time after a new machine is sequenced, all previously established sequences are locally reoptimized. Both the bottleneck identification and the local re-optimization procedures are based on repeatedly solving a certain one-machine scheduling problem that is a relaxation of the original problem. The method of solving the one-machine problems is not new, although they have speeded up considerably the time required for generating these problems. Instead, the main contribution of their approach is the way to use this relaxation to decide upon the order in which the machines should be sequenced. This is based on the classic idea of giving priority to bottleneck machines.

2.3 Genetic Representations for JSP

Because of the existence of the precedence constraints of operations, JSP is not as easy as the *traveling salesmen problem* (TSP) to find a nature representation. There is no good representation with a system of inequalities for the precedence constraints. Therefore, the penalty approach is not easily applied to handle such kind of constraints. Orvosh and Davis (1994) have shown that, for many combinatorial optimization problems, it is relatively easy to repair an infeasible or illegal chromosome and the repair strategy did indeed surpass other strategies such as rejecting strategy or penalizing strategy. Most GA and JSP researchers prefer to take repairing strategy to handle the infeasibility and illegality. A very important issue in building a genetic algorithm for a job-shop problem is to devise an appropriate representation of solutions together with problem-specific genetic operations in order that all chromosomes generated in either initial phase or evolutionary process will produce feasible schedules. This is a crucial phase that conditions all the subsequent steps of genetic algorithms. During the last few years, the following six representations for job-shop scheduling problem have been proposed :

- Operation-based representation
- Job-based representation
- Preference list-based representation
- Priority rule-based representation
- Completion time-based representation
- Random key-based representation
- These representations can be classified into the following two basic encoding approaches :
- Direct approach
- Indirect approach

In the direct approach, a schedule (the solution of JSP) is encoded into a chromosome and genetic algorithms are used to evolve those chromosomes to find a better schedule. The representations, such as operation-based representation, job-based representation, job pair relation-based representation, completion time-based representation, and random keys representation belong to this class. In the indirect approach, such as priority rule-based representation, a sequence of dispatching rules for job assignment, but not a schedule, is encoded into a chromosome and genetic algorithms are used to evolve those chromosomes to find out a better sequence of dispatching rules. A schedule then is constructed with the sequence of dispatching rules. Preference list-based representation, priority rule-based representation, disjunctive graph-based representation, and machine-based representation belong to this class. The detail explanations are shown in Gen *et al.* (2008).

2.4 Experiments on Benchmark Problems

Fisher and Thompson proposed three well-known benchmarks for job-shop scheduling problems in 1963 (Fisher and Thompson, 1963); since then, researchers in operations research have tested their algorithms on these problems. Most GA and JSP researchers used these benchmarks to test the performance of their algorithms. <Table 2> summarizes the experimental results. It lists problem name, problem dimension (number of jobs × number of operations), the *best known solution* (BKS), the solution obtained by different algorithms, where Dorndorf1 stands for the hybrid approach of GA with Giffler and Thompson heuristic and Dorndorf2 for the hybrid one of GA with bottleneck shifting heuristic proposed by Dorndorf and Pesch.

Table 2. Fisher and Thompson's benchmark problems

	FT06 (6×6) (time unit)	FT10 (10×10) (time unit)	FT20 (20×5) (time unit)
BKS	55	930	1165
Gonçalves *et al.* (2005)	55	930	1165
Aiex *et al.* (2003)	55	930	1165
Binato *et al.* (2002)	55	938	1169
Wang and Zheng, (2001)	55	930	1165
Gonçalves and Beirão, (1999)	55	936	1177
Nowicki and Smutnicki, (1996)	55	930	1165
Croce *et al.* (1995)	55	946	1178
Cheng *et al.* (1995)	55	948	1196
Dorndorf1 *et al.* (1995)	55	960	1249
Dorndorf2 *et al.* (1995)	55	938	1178
Gen *et al.* (1994)	55	962	1175
Fang *et al.* (1993)	-	949	1189
Yamada *et al.* (1992)	55	930	1184
Paredis *et al.* (1992)	-	1006	-
Nakano *et al.* (1991)	55	965	1215

3. Assembly Line Balancing Model

An *assembly line* (AL) is a manufacturing process consisting of various tasks in which interchangeable parts are added to a product in a sequential manner at a station to produce a finished product. Assembly lines are the most commonly used method in a mass production environment, because they allow the assembly of complex products by workers with limited training, by dedicated machines and/or by robots.

The installation of an assembly line is a long-term decision and usually requires large capital investments. Therefore, it is important that an AL is designed and balanced so that it works as efficiently as possible. Most of the work related to the ALs concentrate on the ALB. The ALB model deals with the allocation of the *tasks* among *stations* so that the precedence relations are not violated and a given objective function is optimized.

Besides balancing a newly designed assembly line, an existing assembly line has to be re-balanced periodically or after certain changes in the production process or the production plan. Because of the long-term effect of balancing decisions, the objective functions have to be carefully chosen while considering the strategic goals of the enterprise.

3.1 Classification of ALB Models

Based on the model structure, ALB models can be classified into two groups (see <Figure 2>). While, the first group (Scholl, 1999), (Becker and Scholl, 2006) includes *single-model assembly line balancing* (smALB), *multi-model assembly line balancing* (muALB), and *mixed-model assembly line balancing* (mALB); the second group (Baybars, 1986) includes *simple assembly line balancing* (sALB) and *general assembly line balancing* (gALB). The smALB model involves only one product. The muALB model involves more than one product produced in batches. The mALB refers to assembly lines which are capable of producing a variety of similar product models simultaneously and continuously (not in batches). Additionally, sALB, the simplest version of the ALB model and the special version of smALB model, involves production of only one product with features such as paced line with fixed cycle time, deterministic independent processing times, no assignment restrictions, serial layout, one

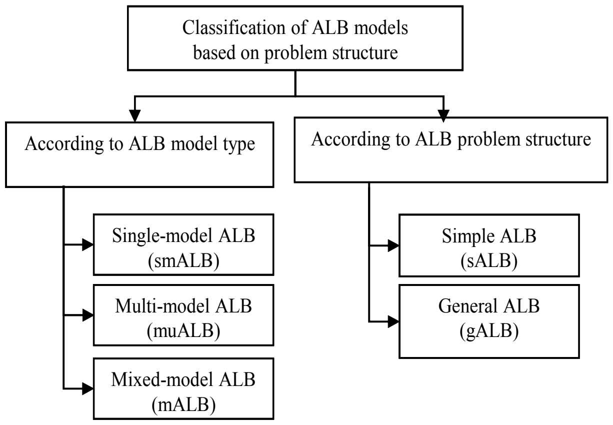
Classification of ALB models based on problem structure
According to ALB model type
According to ALB problem structure
Single-model ALB (smALB)
Multi-model ALB (muALB)
Mixed-model ALB (mALB)
Simple ALB (sALB)
General ALB (gALB)

Figure 2. Classification of assembly line balancing models

sided stations, equally equipped stations and fixed rate launching. The gALB model includes all of the models that are not sALB, such as balancing of mixed-model, parallel, u-shaped and two sided lines with stochastic dependent processing times; thereby more realistic ALB models can be formulated by gALB.

Additionally, several versions of ALB problems arise by varying the objective function (Scholl, 1999). *Type-F* is an objective independent problem which is to establish whether or not a feasible line balance exists. *Type-1* and *Type-2* have a dual relationship; the first one tries to minimize the number of stations for a given cycle time, and the second one tries to minimize the cycle time for a given number of stations. *Type-E* is the most general problem version, which tries to maximize the line efficiency by simultaneously minimizing the cycle time and a number of stations. Finally, Type-3, 4 and 5 correspond to maximization of workload smoothness, maximization of work relatedness and multiple objectives with Type-3 and Type-4, respectively (Kim *et al*., 1996).

3.2 Research on ALB Models

Since the ALB model was first formulated by Helgeson *et al.* (1954), many solution approaches have been proposed. Several optimum seeking methods, such as *linear programming* (Salveson, 1955), *integer programming* (Bowman, 1960), *dynamic programming* (Held *et al.*, 1963) and *branch-and-bound approaches* (Jackson, 1956) have been employed to deal with ALB. However, none of these methods has proven to be of practical use for large problems due to their computational inefficiency. Since, ALB model falls into the NP-hard class of combinatorial optimization problems (Karp, 1972), in recent years, to provide an alternative to traditional optimization techniques, numerous research efforts have been directed towards the development of heuristics (Dar-El, 1973) and meta-heuristics. While heuristic methods generating one or more feasible solutions were mostly developed until mid 90s; meta-heuristics such as tabu search (Scholl, 1996), simulated annealing (Suresh, 1994), genetic algorithms (Falkenauer and Delchambre, 1992) and ant colony optimization (Bautista and Pereira, 2002) have been the focus of researchers in the last decade.

For more information, the reader can refer to several review studies, *i.e.* Baybars (1986) that surveys the exact (optimal) methods, Talbot *et al.* (1986) that compare and evaluate the heuristic methods developed, Ghosh and Gagnon (1989) that present a comprehensive review and analysis of the different methods for design, balancing and scheduling of assembly systems, Erel and Sarin (1998) that present a comprehensive review of the procedures for smALB, muALB and mALB models, Rekiek *et al.* (2002) that focus on optimization methods for the line balancing and resource planning steps of assembly line design, Scholl and Becker (2006) that present a review and analysis of exact and heuristic solution procedures for sALB, Becker and Scholl (2006) that present a survey on problems and methods for gALB with features such as cost/profit oriented objectives, equipment selection/ process alternatives, parallel stations/tasks, u-shaped line layout, assignment restrictions, stochastic task processing times and mixed model assembly lines, Rekiek and Delchambre (2006) that focus on solutions methods for solving sALB, and Ozmehmet Tasan and Tunali (2008) that present a comprehensive review of GAs approaches used for solving various ALB models.

Among the meta-heuristics, the application of *Genetic Algorithms* (GAs) received a considerable attention from the researchers, since it provides an alternative to traditional optimization techniques by using directed random searches to locate optimum solutions in complex landscapes and it is also proven to be effective in various combinatorial optimization problems. GAs are powerful and broadly applicable stochastic search and optimization techniques based on principles from evolutionary theory (Gen and Cheng, 2000).

Falkenauer and Delchambre (1992) were the first to solve ALB with GAs. Following Falkenauer and Delchambre (1992), application of GAs for solving ALB model was studied by many researchers, e.g., (Kim *et al.*, 1996), (Leu *et al.*, 1994), (Noorul *et al.*, 2006). However, most of the researchers focused on the simplest version of the problem, with single objective and ignored the recent trends, *i.e.*, mixed-model production, u-shaped lines, robotic lines and etc, in the complex assembly environments, where ALB models are multiobjective in nature (Ozmehmet Tasan and Tunali, 2008).

3.3 Mathematical Formulation of ALB Models

The basic version of the ALB model is the simple assembly line balancing (sALB) model. The simple assembly line is a single-model assembly line that is capable of producing only one type of product. The simple assembly line can be defined by the following assumptions (Scholl, 1999, Baybars, 1986) :

A1 : The line is used to assemble one homogeneous product in mass quantities.
A2 : The line is serial, paced line with fixed cycle time and there are no feeder or parallel subassembly lines.
A3 : The processing times of tasks are deterministic.
A4 : All stations are equally equipped with respect to machines and workers.
A5 : A task cannot be split among two or more stations.
A6 : There are no assignment restrictions besides the precedence constraints.
A7 : All stations can process any one of the tasks and all have the same associated costs.
A8 : The processing time of a task is independent of the station and furthermore, they are not sequence dependent.

Among the family of ALB models, the most well-known and well-studied is certainly the sALB model. Although it might be far too constrained to reflect the complexity of real-world line balancing, it nevertheless captures its main aspects and is rightfully regarded as the core model of ALB. In fact, vast varieties of more general problems are direct sALB model extensions or at least require the solution of sALB instances in some form. In any case, it is well suited to explain the basic principles of ALB and introduce its relevant terms.

A simple AL capable of producing only one type of product consists of *stations* ($i = 1, \cdots, m$) arranged along a conveyor belt or a similar mechanical material handling equipment. The workpieces (jobs) are consecutively launched down the line and are moved from station to station. At each station, certain tasks are repeatedly performed regarding the *cycle time* (maximum or average time available for each workcycle). The decision problem of optimally partitioning, *i.e.*, *balancing*, the assembly work among the stations with respect to a given objective function is known as sALB problem.

Manufacturing a product on an assembly line requires partitioning the total amount of work into a set of elementary operations named *tasks* $j = \{1, \cdots, n\}$. Performing a task j takes certain task time t_j and requires certain equipment of machines and/or skills of workers. Due to technological and organizational conditions *precedence constraints* between the tasks have to be observed. These elements can be summarized and visualized by a *precedence network graph*. It contains a node for each task, node weights for the task processing times and arcs for the precedence constraints.

<Table 3> presents the data set for an example sALB model, which contains 12 tasks ((Runarsson, 1999)). Using this data set, the precedence graph in <Figure 3> is constructed. The precedence graph contains 12 nodes for tasks, node weights for task processing times and arcs for orderings. For example, the processing time for task 6 is 5 time units. For the processing of task 6, tasks 3 and 5 (direct predecessors) and tasks 1, 2 and 4 (indirect predecessors) must be completed. Likewise, task 6 must be completed before tasks 7, 9 and 10 (direct successors), and tasks 8, 10, and 12 (indirect successors) start processing.

Table 3. Data set of the sALB model

Task j	Suc(j)	Task time t_j
1	{2, 4}	5
2	{3}	3
3	{6}	4
4	{5}	3
5	{6}	6
6	{7, 9, 10}	5
7	{8}	2
8	{12}	6
9	{12}	1
10	{11}	4
11	{12}	4
12	{}	7

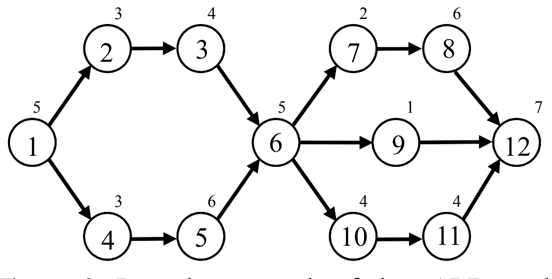

Figure 3. Precedence graph of the sALB model

Any type of sALB model consists in finding a feasible *line balance*, *i.e.*, an assignment of each task to a station such that the precedence constraints are fulfilled. The set of tasks S_i assigned to a station i (= 1, ⋯, m) constitutes its *station load*, the cumulated task time

$$t(S_i) = \sum_{j \in S_i} t_j \tag{4}$$

is called *station time*. When a fixed common *cycle time* c_T is given, a line balance is feasible only if the station time of neither station exceeds c_T. In case of $t(S_i) < c_T$, the station i has an idle time of (c_T-$t(S_i)$) time units in each cycle.

Using the precedence graph in <Figure 3>, a feasible line balance with cycle time 11 time units and 6 stations can be constructed by the station loads S_1 = {1, 2}, S_2 = {3, 4}, S_3 = {5, 6}, S_4 = {7, 9, 10, 11}, S_5 = {8}, S_6 = {12} (see <Figure 4>). While no idle time occurs in stations 3 and 4, stations 1, 2, 5 and 6 show idle times of 3, 4, 5 and 4 time units, respectively.

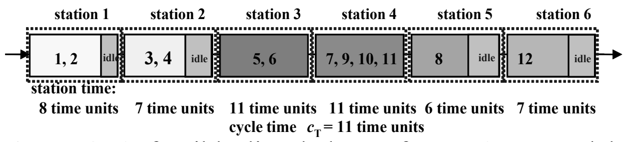

Figure 4. A feasible line balance for a sALB model

In order to formulate the mathematical model, the following indices, parameters and decision variable are introduced :

Notations

Indices

i	: index of the station (i = 1, ⋯, m)
j, k	: index of the task (j, k = 1, ⋯, n)

Parameters

m	: number of stations actually employed
M	: maximum number of stations available
n	: number of tasks
c_T	: cycle time of the assembly line
t_j	: processing time of task j
S_i	: the set of tasks assigned to station i
T_S	: a task sequence
Suc(j)	: the set of direct successors of task j
Pre(j)	: the set of direct predecessors of task j
u_i	: *utilization* of the station i

$$u_i = \frac{1}{\max_{1 \le i \le m}\{t(S_i)\}} t(S_i) \tag{5}$$

$\bar{u}$: *average utilization* of total stations

$$\bar{u} = \frac{1}{m}\sum_{i=1}^{m} u_i \tag{6}$$

Decision Variables

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

The mathematical model for the sALB Type-1 can be stated as follows :

Mathematical Model

$$\max \quad E = \frac{1}{m\, c_T}\sum_{j=1}^{n} t_j x_{ij} \tag{7}$$

$$\min \quad m = \sum_{i=1}^{M} \max_{1 \le j \le n}\{x_{ij}\} \tag{8}$$

$$\min \quad V = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(u_i - \bar{u})^2} \tag{9}$$

$$\text{s.t.} \quad \sum_{i=1}^{M} x_{ij} = 1 \quad \forall j \tag{10}$$

$$\sum_{i=1}^{M} i\, x_{ik} \le \sum_{i=1}^{M} i\, x_{ij} \quad \forall j,\ k \in \text{Pre}(j) \tag{11}$$

$$t(S_i) = \sum_{j \in S_i} t_j = \sum_{j=1}^{n} t_j x_{ij} \le c_T \quad \forall i \tag{12}$$

$$x_{ij} = 0 \text{ or } 1 \quad \forall i,\ j \tag{13}$$

In this mathematical model, the first objective (Eq. 7) of the model is to maximize the *line efficiency*. The second objective (Eq. 8) is to minimize the *number of stations* actually employed. The third objective (Eq. 9) of the model is to minimize the *variation of workload*. The constraints given in equations (10)~(13) are used to formulate the general feasibility of the problem. The constraint given in equation (10) states that each task must be assign to one and only one station. Inequity (11) represents the precedence constraints and it states that the direct predecessor of task j must be assign to a station, which is in front of or the same as the station that task j is assigned in. This constraint stresses that if a task is assigned to a station, then the predecessor of this task must be already assigned to a station. Inequity (12) denotes that the available time at each station should be less than or equal to the given cycle time. Constraint given in equation (13) represents the usual integrity restriction.

3.4 Genetic Representation for ALB Models

The primary issue in applying GA to problem is to convert the information of ALB model into a genetic representation form. Up to now, several genetic representations, *i.e.*, task-based, embryonic, workstation-based, grouping-based, and heuristic-based have been proposed, each having pros and cons concerning the type of applicable genetic operators.

The chromosome representation schemes are named in order to suit the characteristics of ALB model (Ozmehmet Tasan and Tunali, 2008). These representation schemes can be classified as follows :

Task-based Encoding : The chromosomes are defined as feasible precedence sequences of tasks (Miltenburg, 2002), (Sabuncuoglu *et al.*, 2000). The length of the chromosome is defined by the number of tasks. For example, the task based representation of the solution is illustrated in <Figure 5(a)>. In order to calculate the fitness of a task based chromosome, additional operations, which assign the tasks to workstations according to the task sequence in the chromosome, are needed. Task-based representation is the most appropriate representation for ALB Type-1 models, since Type-1 models consider the minimization of stations as an objective function.

Embryonic Encoding : Embryonic chromosome representation that was proposed by Brudaru and Valmar (2004) is actually a special version of the task based chromosome. The only difference between the two is that the embryonic representation of a solution considers the subsets of solutions rather than the individual solutions. During the generations, the embryonic chromosome evolves through a full length solution. Therefore, the chromosome length varies throughout the generations. The length is initially defined by a random number and then increases until it reaches the number of tasks. <Figure 5(b)> illustrates an example of embryonic representation

Workstation-based Encoding : The chromosome is defined as a vector containing the labels of the stations to which the tasks are assigned (Anderson and Ferris, 1994), (Kim *et al.*, 2000). The chromosome length is defined by the number of tasks. For example, the workstation based representation of the solution is illustrated in <Figure 5(c)>, where task 4 is assigned to station 3. This kind of chromosome representation scheme is generally used for ALB Type-2 models.

Grouping-based Encoding : This type of representation was proposed by Falkenauer and Delchambre (1992) especially for grouping problems, *i.e.*, ALB Type-1 models. The authors stated that the workstation-based representation, which is object oriented, is not suitable for ALB Type-1 models. In grouping-based representation, the stations are represented by augmenting the workstation-based chromosome with a group part. The group part of the chromosome is written after a semicolon to list all of the workstations in the current solution (see <Figure 4(d)>). The length of the chromosome varies from solution to solution. As is seen in <Figure 5(d)>, the first part is the same as in workstation-based chromosome. The difference comes from the grouping part, which list all the stations, *i.e.*, 1, 2, 3, 4, 5, and 6.

Heuristic-based (Indirect) Encoding : This type of representation scheme represents the solutions in an indirect manner. In Goncalves and De Almeida (2002), and Bautista *et al.* (2000), the authors first coded the priority values of the tasks (or a sequence of priority rules), then they applied these rules to the problem to generate the solutions. The chromosome length is defined by the number of heuristics. For example, <Figure 5(e)> shows an example chromosome having seven different heuristics, which are used in the sequence of H_1, H_2, H_5, H_4, H_7, H_6 and H_3 to assign the tasks to the workstations.

3.5 Robotic Assembly Line Balancing Model

In the past decades, robots have been extensively used in assembly lines as called *robotic assembly lines* (rALs). An assembly robot can work 24 hours a day without worries or fatigue. Goals for implementation of robotic assembly lines include high productivity, quality of product, manufacturing flexibility, safety, decreasing demand of skilled labor, and so on. Different robot types may exist at the assembly facility. Each robot type may have different capabilities and efficien-

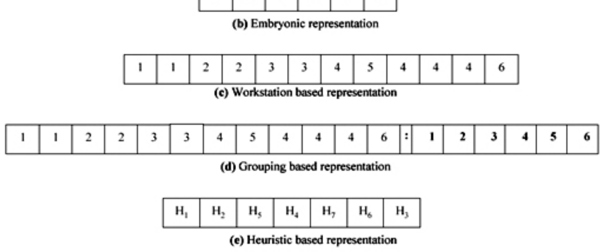
1 2 4 3 5 6 7 10 9 11 8 12
(a) Task based representation
1 3 2 5 4 6
(b) Embryonic representation
1 1 2 2 3 3 4 5 4 4 4 6
(c) Workstation based representation
1 1 2 2 3 3 4 5 4 4 4 6 : 1 2 3 4 5 6
(d) Grouping based representation
H1 H2 H5 H4 H7 H6 H3
(e) Heuristic based representation

Figure 5. Chromosome representation schemes used in ALB models

cies for various elements of the assembly tasks. Usually, specific tooling is developed to perform the activities needed at each station. Such tooling is attached to the robot at the station. In order to avoid the time waste required for tool change, the design of the tooling can take place only after the line has been balanced. Hence, to allocate the best fitting robot for each station is critical for the performance of rALs.

Unlike manual assembly lines, where actual processing times for performing task vary considerably and optimal balance is rather of theoretical importance, the performance of rALs depends strictly on the quality of its balance. As extended from sALB, r*obotic assembly line balancing* (rALB) is also NP-hard.

Rubinovitz and Bukchin (1993) were the first to formulate the rALB model as one of the allocating equal amounts of work to the stations on the line while assigning the most efficient robot type from the given set of available robots to each workstation. Their objective is to minimize the number of workstations for a given cycle time. Following, the authors (Rubinovitz and Bukchin, 1991) presented a branch and bound algorithm for the problem. Bukchin and Tzur (Bukchin and Tzur, 2000) treat the problem with objective to minimize the total equipment cost, given a predetermined cycle time, where they developed an exact branch and bound algorithm, meanwhile a branch-and-bound-based heuristic procedure is suggested for large problems. Tsai and Yao (1993) proposed a heuristic approach for the design of a flexible robotic assembly line which produces a family of products. Kim and Park (1995) extended the problem by considering additional constraints, *i.e.*, due to limited space to store the parts and tools, restrictions for the joint assignment of tasks to stations are imposed. They proposed a mathematical formulation and a cutting plane procedure for this extension of the problem.

Khouja *et al*. (2000) suggested statistical clustering procedures to design robotic assembly cells. Nicosia *et al*. (2002) considered the problem of assigning operations to an ordered sequence of non-identical workstations under the constraints of precedence relationships and a given cycle time. The objective is to minimize the cost of the workstations. This formulation is very similar to the rALB problem.

The aforementioned rALB works have assumed that the cycle time is predetermined, and aimed at minimizing the number of workstations or the cost of the assembly systems. Hence, these works are of the call of the rALB Type-1 model. Levintin *et al*. (2006) dealt with a rALB Type-2 model, in which different robots may be assigned to the assembly line tasks, and each robot needs different assembly times to perform a given task due to its capabilities and specialization. The objective is to maximizing the production rate of the line. Two genetic algorithms are presented to solve the rALB Type-2 model.

Since the number of stations is determined by the number of robots in a rAL, in this section we will consider rALB Type-2 model. This model is usually present when changes in the production process of a product take place. For example, a new product is introduced for assembly. In this case, the rAL has to be reconfigured using the present resources (such as robots) so as to improve it efficiency for the new production process. The model concerns how to assign the tasks to stations and how to allocate the available robots for each station in order to minimize the cycle time under the constraint of precedence relationships. In this case, the number of stations of the line and the available robots may remain fixed. The following assumptions are stated to clarify the setting in which the rALB model arises :

A1 : The precedence relations among assembly tasks are known and constant.
A2 : The processing times of tasks are deterministic and dependent on the assigned robot.
A3 : A task cannot be split among two or more stations.
A4 : There are no limitations on assignment of task or robots to any station besides the precedence constraints. In case a task can not be processed on a robot, the processing time of the task on the robot is set to very high.
A5 : A single robot is assigned to each station.
A6 : Material handling, loading and unloading times, as well as set-up and tool changing times are negligible, or are included in the activity times. This assumption is realistic on a sALB that works on the single product for which it is balanced. Tooling on such rAL is usually designed such that tool changes are minimized within a station. If tool change or other type of set-up activity is necessary, it can be included in the activity time, since the transfer lot size on such line is of a single product.
A7 : The number of stations is determined by the number of robots, since the problem aims to maximize the productivity by using all robots at

hand.

A8 : The line is balanced for a single product.

3.5.1 Mathematical Formulation of rALB Models

The rALB Type-2 model focuses on the assignment of tasks to stations and to allocate robot for each station with the objective of minimum cycle time given the number of stations as the available robots.

<Table 4> presents the data set for an example rALB model, which contains 10 tasks, 4 robots assigned to 4 stations and the processing time of each task processed by each robot. Using this data set, the precedence graph in <Figure 6> is constructed.

Table 4. Data set of the rALB model

Task i	Suc(i)	R_1	R_2	R_3	R_4
1	{4}	17	22	19	13
2	{4}	21	22	16	20
3	{5}	12	25	27	15
4	{6}	29	21	19	16
5	{10}	31	25	26	22
6	{8}	28	18	20	21
7	{8}	42	28	23	34
8	{10}	27	33	40	25
9	{10}	19	13	17	34
10	-	26	27	36	26

The precedence graph contains 10 nodes for tasks and arcs for orderings. For example, the processing time for task 7 by robot 3 is 23 time units. For the processing of task 8, tasks 6 and 7 (direct predecessors) and task 4 (indirect predecessor) must be completed. Likewise, task 8 must be completed before task 10 (direct successor) starts processing.

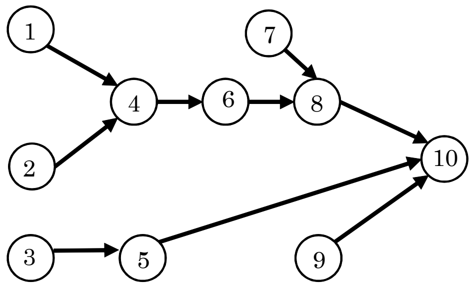

Figure 6. Precedence graph of the rALB model

Using the precedence graph in <Figure 6>, a feasible line balance with cycle time 85 time units and 4 stations can be constructed by the station loads S_1 = {2, 7}, S_2 = {3, 5}, S_3 = {1, 4, 6}, S_4 = {9, 8, 10} (see <Figure 3>). While no idle time occurs in station 4, but stations 1, 2, and 3, 4 have very long idle times, which mean that the line is not effectively balanced.

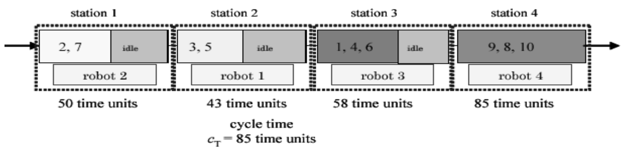

Figure 7. A feasible line balance for a rALB model

The mathematical model for the rALB Type-2 can be stated as follows :

$$\min \quad c_{\mathrm{T}} = \max_{1 \le k \le m} \left\{ \sum_{i=1}^{n} \sum_{l=1}^{m} t_{il} x_{ik} y_{kl} \right\} \tag{14}$$

In this mathematical model, the objective (Eq. 14) is to minimize the cycle time (c_{T}). In this equation, I, j is index of tasks, k is index of stations, l is index of robots, m is number of stations (robots), n is number of tasks, t_{il} is processing time of the task i by robot l. and x_{ik}, y_{lk} represents the usual integrity restriction. If task j is assigned to station k, x_{ik} = 1, otherwise, x_{ik} = 0. If robot l is assigned to station k, y_{lk} = 1, otherwise, y_{lk} = 0.

3.5.2 Hybrid GA for rALB Models

To develop a genetic representation for the rALB model, there are three main phases :

Phase 1 : Creating a task sequence
- step 1.1 : Order encoding for task sequence by randomly generating a list of tasks.
- step 1.2 : Reordering the tasks to a feasible task sequence that satisfies the precedence constraints.
- step 1.3 : Breakpoint decoding to assign the tasks into each station by breakpoint decoding.

Phase 2 : Assigning robots to each station
- step 2.1 : Order encoding for robot assignment by randomly assign the robots to each station.

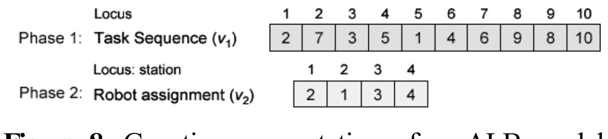

Figure 8. Genetic representation of a rALB model

step 2.2 : Breakpoint decoding to assign the robots into each station.

Phase 3 : Designing a schedule

step 3.1 : Creating a schedule for the assembly line.

step 3.2 : Drawing a Gantt chart for this schedule.

A solution of the rALB model can be represented by two integer vectors, *i.e.*, v_1 and v_2. Task sequence vector (v_1), which contains a permutation of assembly tasks, ordered according to their technological precedence sequence, and robot assignment vector (v_2). The solution representation method can be visually illustrated as in <Figure 8>.

<Figure 9> illustrates the process of breakpoint decoding procedure on a chromosome. Here, a cycle time is said to be feasible if all the tasks can be allocated to the stations by allowing as many tasks as possible for each station under the constraint of the cycle time.

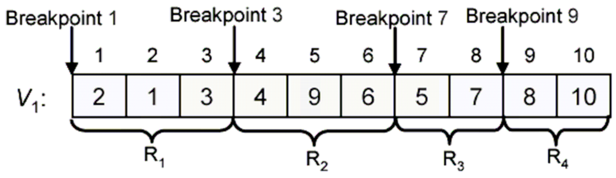

Figure 9. Breakpoint decoding

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

Schedule $S = (j, R_i, t_j)$:

S = {(t_1, R_1, 0～17), (t_2, R_1, 17～38), (t_3, R_1, 38～50), (t_4, R_2, 50～71), (t_5, R_3, 102～128), (t_6, R2, 71～89), (t_7, R_3, 128～151), (t_8, R_4, 151～176), t_9, R_2, 89～102), (t_{10}, R_4, 176～202)}

<Figure 10> and <Figure 11> show the Gantt charts for one unit and three units of product, respectively.

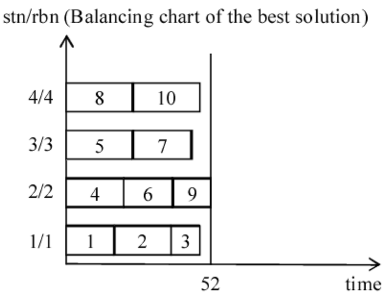

Figure 10. The balance chart of the best solution

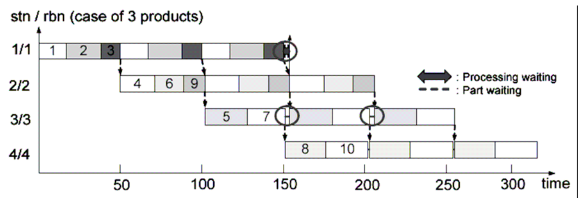

Figure 11. Gantt chart for the rALB model(three units of product)

Robot Assignment Search : The neighborhood can be defined as the set of solutions obtainable from an initial solution by some specified perturbation. A robot assignment neighbor solution is generated by exchanging the robot assigned on the critical station with another robot. The robot assignment search is the local search which works over robot assignment neighborhood.

Let $N(i)$ denote the set of machine assignment neighborhood of solution i. The enlarged two-pace machine assignment neighborhood is defined as the union of the neighborhood of each robot assignment neighbor of solution i (see <Figure 12>).

$$N^2(i) = \cup_{j \in N(i)} N(j)$$

During the robot assignment search, the local search will implement over two-pace neighborhood when it reaches the local optima of one-pace neighborhood, and is called two-pace robot assignment search. During the robot assignment search, when the local optima of two-pace robot assignment neighborhood is reached, the neighbors of the two-pace local optima help the robot assignment search escape from the local optima.

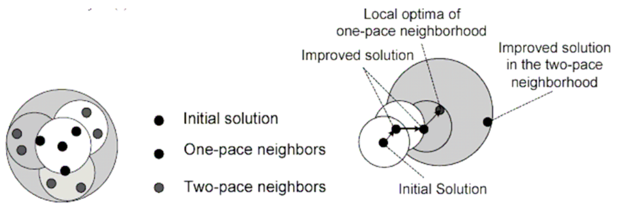

Figure 12. Illustration of the neighborhood

3.5.3 Numerical Experiment

In this study, eight representative precedence graphs, which are widely used in the sALB Type-1 models (Scholl, 1993), are used. These precedence graphs contain 25～297 tasks. From each precedence graph, four different rALB Type-2 models are generated by using different WEST ratios : 3, 5, 7, 10, 15. WEST ratio, as defined by Dar-El (1973), measures the average number of activities per station. This measure indicates the expected quality of achievable solutions and complexity of the problem. For each problem, the number of station is equal to the number of robots, and each task can be processed on any robot.

In this study, a local search method is proposed to enhance the search ability of GA. The local search method, *i.e.*, *Robot Assignment Search*, is based on critical stations in order to improve their effectiveness and efficiency.

To evaluate the performance of the hGA, these 32 test instances are solved using the hGA approach. Additionally, the two algorithms proposed by Levitin

Table 5. The results of the computational experiments

The Problem			Cycle time(c_t)		
No. of tasks	No. of stations	WEST ratio	Levitim *et al*'s Recursive	Levitim *et al*'s Conse-cutive	Hyhrid GA approach
25	3	8.33	518	**503**	**503**
	4	6.25	351	330	**327**
	6	4.17	343	234	**213**
	9	2.78	138	125	**123d**
35	4	8.75	551	450	**449**
	5	7.00	385	352	**344**
	7	5.00	250	**222**	**222**
	12	2.92	178	120	**113**
53	5	10.60	903	565	**554**
	7	7.57	390	342	**320**
	10	5.30	35	251	**230**
	14	3.79	243	166	**162**
70	7	10.00	546	490	**449**
	10	7.00	313	287	**272**
	14	5.00	231	213	**204**
	19	3.68	198	167	**154**
89	8	11.13	638	505	**494**
	12	7.42	455	371	**370**
	16	5.56	292	246	**236**
	21	4.24	277	209	**205**
111	9	12.33	695	586	**557**
	13	8.54	401	339	**319**
	17	6.53	322	**257**	**257**
	22	5.05	265	209	**192**
148	10	14.80	708	638	**600**
	14	10.57	537	441	**427**
	21	7.06	404	325	**300**
	29	5.10	249	210	**202**
297	19	15.63	1129	674	**646**
	29	10.24	571	444	**430**
	38	7.82	442	348	**344**
	50	5.94	363	275	**256**

et al. (2006) are also used to solve the 32 problems. <Table 4> presents the performance of the hGA approach for rALB Type-2 model.

From the result, it can be states that the hGA approach performs better than Levitin *et al.*'s two algorithms along with increasing the scale for the problems. The computational experiments show that this algorithm is computationally efficient and effective to find the optimal solution. Additionally, <Figure 13> illustrates the evolutionary process of the three algorithms on problem 148-21.

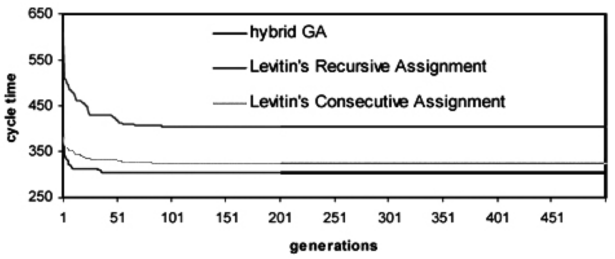

Figure 13. Evolutionary process of problem 148-21

4. Manufacturing and Logistics Model

4.1 Manufacturing Models

Most of the literature on the shop scheduling problem concentrates on the JSP case (Gen and Cheng, 1997, 2000). The *flexible job-shop scheduling Problem* (fJSP) is expanded from the traditional Job-shop Scheduling Problem, which possesses wider availability of machines for all the operations. The fJSP recently captured the interest of many researchers. The first paper that addresses the fJSP was given by Brucker and Schlie (1990), who proposed a polynomial algorithm for solving the fJSP with two jobs, in which the machines able to perform an operation have the same processing time. For solving the general case with more than two jobs, two types of approaches have been used : hierarchical approaches and integrated approaches. The first was based on the idea of breaking down the original problem in order to reduce its complexity. Brandimarte (1993) was the first to use this breaking down for the fJSP. He solved the assignment problem using some existing dispatching rules and then focused on the resulting job shop subproblems, which are solved using a tabu search heuristic. Mati *et al.* (2001) proposed a greedy heuristic for simultaneously dealing with the assignment and the sequencing subproblems of the flexible job shop model. The advantage of Mati's heuristic is its ability to take into account the assumption of identical machine. Kacem *et al.* (2002) came to use GA to solve fJSP, and adapted two approaches to solve jointly the assignment and JSP (with total or partial flexibility). The first is the approach by localization. It makes it possible to solve the problem of resource allocation and build an ideal assignment mode (assignments schemata). The second is an evolutionary approach controlled by the assignment model, and applying GA to solve the fJSP. Wu and Weng (2005) considered the problem with job earliness and tardiness objectives, and proposed a multiagent scheduling method. Xia and Wu (2005) treated this problem with a hybrid of *particle swarm optimization* (PSO) and *simulated annealing* (SA) as a local search algorithm. Zhang and Gen (2005) and Gen and Zhang (2006) proposed a multistage operation-based genetic algorithm to deal with the fJSP problem from the point view of dynamic programming.

In order to obtain farthest profit, manufacturing enterprises tends to improve the ability of responding to the rapidly changing market demands, which require the effective and efficient manufacturing of a variety of products with varying volume (Su *et al.* 2003). Based on the development of information technique and manufacturing conception, IMS is built as the most adaptive and available approach for modern requirement. To get an optimal operation sequence with flexible resource assignment is the main function of IMS, therefore several important issues come to forth of the managers, to find an optimal production planning :

- How to get a minimum execution time (makespan) for responding the emergency or forecasting orders?
- How to deal with different lot size of orders for minimizing the transportation cost by workers or robots?
- How to balance the workload of all the machines in our plants for reducing the work-in-process inventories and operation bottlenecks?
- How to reduce the complex process on transportations between machines in a local plant?

The *integrated operation sequences and resource selection* (iOS/RS) problem is formulated in particular, for the reason that it originally derived from the real production process in manufacturing systems, and approximate to it. For instance, each order consists some operations, while the sequences are not fixed, which in terms of several precedence constraints; lot size and unit load size for different orders are considered in scheduling process, which means the starting time of operations depend on not only the finishing time of preceding operations within the same order but also their finishing time of unit load size.

During the past several years, many researchers have put great effort into the area on integrated process planning and scheduling problem. Tan (2000) reported a briefly review of the research in the process planning and scheduling area and discussed the extent of applicability of various approaches, and also proposed a linearized polynomial mixed integer programming model for this problem in recent research work (Tan, 2004). Dellaert *et al.* (2000) discussed multi-level lot-sizing problem in *material requirements planning* (MRP) systems. They developed a binary encoding genetic algorithm (GA) and design five specific genetic operators to ensure that exploration takes place within the set of feasible solutions. Raa and Aghezzaf (2005) also introduced a robust dynamic planning strategy for lot-sizing problems with stochastic demands in their recent research work. Recently, for improving the flexibility of machine assignment, Kacem *et al.* (2002) proposed a genetic algorithm controlled by the assigned model which is generated by the approach of localization. Najid *et al.* (2002) used SA for optimizing the flexible assignment of machines in fJSP. Lopez and Ramirez (2005) newly describe the design and implementation of a step-based manufacturing information system to share flexible manufacturing resources data.

Anyway, all those research above considered the alternative machines for each operation, and they wanted to apply their model for solving the flexible assignment of various resources (machines). However, there exists a weakness which is fixing all the operation sequence or non-constraint operation sequence. That is they ignore the flexibility especially for orders, which consists some precedence constraints and the corresponding sequences is also alternative.

Especially in recent work by Moon (2004a, 2004b), a GA approach is proposed to solve such kind iOS/RS problem considering the orders with unfixed operation sequence, however, it encoded the chromosome only considering the information of operation sequence, and hybridized with some heuristic strategy for resource selection (by minimum processing time). This approach actually improves the efficiency of convergence and the speed of calculation, but may loose some optimal solution, because they ignore the transition time between different machines. To avoid this limitation, we propose an effective coding approach to formulate the iOS/RS model into two-dimensional. The new idea is built on the basic concept of *multistage decision making* (MSDM) model. We separate all the operations as a set of stages, and in each stage (operation) several alternative states (machines) are offered for selection, hence our job is to make a decision in all the stages for choosing states and get an optimal schedule. For this reason we formulate a multistage *operation-based genetic algorithm* (moGA) proposed by Gen and Zhang (2006), and define the chromosome with two vectors which contains both of the two information, operation sequence and machine selection.

In manufacturing industries, information systems become more important to apply in today's rapidly changing global business environment. Scheduling subsystem is a main module of a manufacturing information system. Until now, the main purpose of scheduling was improvement of equipment operation rate and reduction of cost by the mass production. However, at the present, manufacturers must make plans and schedules considering many kind of customer's demand. Therefore, it is necessary to consider various elements in the optimization of schedule such as not only simple manufacturing scheduling but also logistics, inventory control, and etc. <Figure 14> shows a block diagram of the scheduling system.

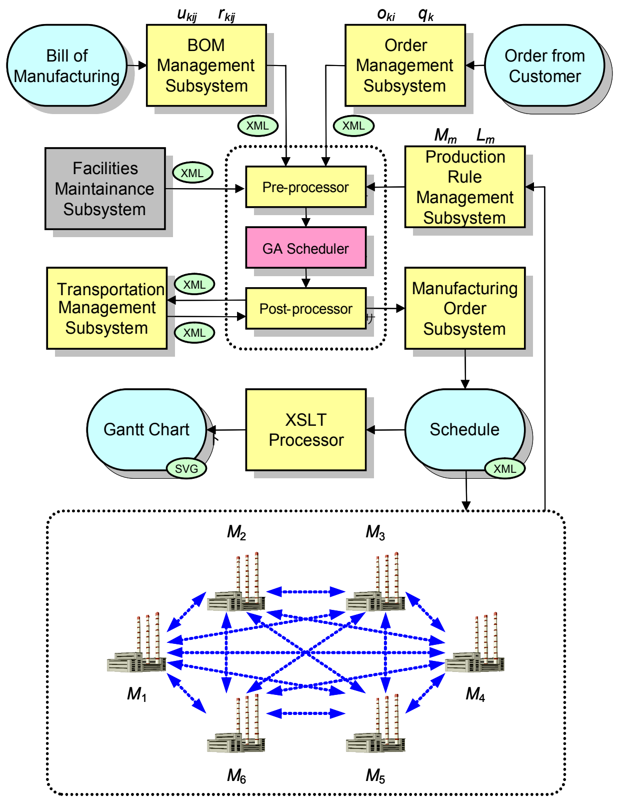

Figure 14. Block diagram of experimental scheduling system

It is difficult to construct the information system including large number of elements as a single system. The enlarged system including various elements becomes difficult to read, change and improve. Moreover, the requirement specification of the system changes in today, while the system has been designed by spending time. PSLX (PSLX Consortium, 2003) proposes the method to construct the APS system by small subsystems called agents, and functions of the system are realized by the cooperation of agents. The standardization of the data format for the communication is important to realize cooperation of agents. Recently, XML (Extensible Markup Language) (W3C, 2004) is widely utilized to exchange data on the Internet. For example, PSLX and MESX (MESX Joint Working Group, 2004) developed the standards based on XML for industries. (Okamoto *et al.*, 2005) proposed a GA-based scheduling agent for APS system using XML, and suggested manufacturing scheduling subsystem is the most important module on APS system. The system has extensible structure according to the merit of GA and XML.

4.2 Logistics Models

In traditional logistics system, the focus of the integration of logistics system is usually on single objective such as minimum cost or maximum profit. For example, Jayaraman and Prikul (2001, Jayaraman and Ross (2003), Yan *et al.* (2003), Syam (2002), Syarif *et al.* (2002), Amiri (2004), Gen and Syarif (2005), Truong and Azadivar (2005), and Gen *et al.* (2006) had considered total cost of logistics as an objective function in their studies. However, there are no design tasks that are single objective problems. The design/planning/scheduling projects are usually involving tradeoffs among different incompatible goals. Recently, multi objective optimization of logistics has been considered by different researchers in literature. Sabri and Beamon (2000) developed an integrated multi-objective supply chain model for strategic and operational supply chain planning under uncertainties of product, delivery and demand. While cost, fill rates, and flexibility were considered as objectives, and constraint method had been used as a solution methodology. Chan and Chung (2004) proposed a multi-objective genetic optimization procedure for the order distribution problem in a demand driven logistics. They considered minimization of total cost of the system, total delivery days and the equity of the capacity utilization ratio for manufacturers as objectives. Chen and Lee (2004) developed a multi-product, multi-stage, and multi-period scheduling model for a multi-stage logistics with uncertain demands and product prices

As objectives, fair profit distribution among all participants, safe inventory levels and maximum customer service levels, and robustness of decision to uncertain demands had been considered, and a two-phased fuzzy decision-making method was proposed to solve the problem. Erol and Ferrell (2004) proposed a model that assigning suppliers to warehouses and warehouses to customers. They used a multi-objective optimization modeling framework for minimizing cost and maximizing customer satisfaction. Guillen *et al.*, (2005) formulated the logistics network model as a multi-objective stochastic mixed integer linear programming model, which was solved by e-constraint method, and branch and bound techniques. Objectives were SC profit over the time horizon and customer satisfaction level. Chen *et al.* (2005) developed a hybrid approach based on genetic algorithm and *analytic hierarch process* (AHP) for production and distribution problems in multi-factory supply chain models. Operating cost, service level, and resources utilization had

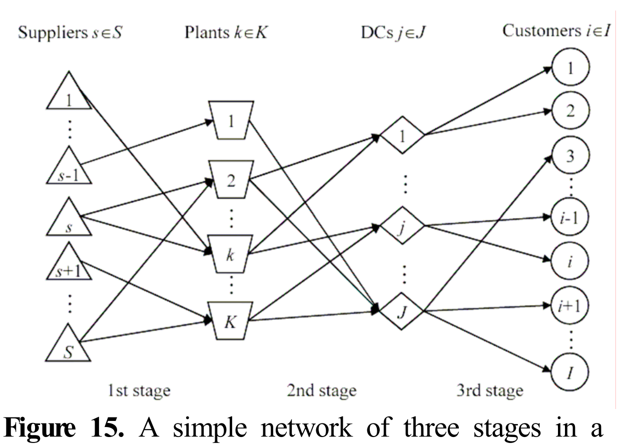

Figure 15. A simple network of three stages in a supply chain network

been considered as objectives in their study. Altiparmak *et al.* (2006, 2007) developed multiobjective genetic algorithms for a single-source, single and multi-product, multi-stage SCN design problems. The studies reviewed above have found a Pareto-optimal solution or a restrictive set of Pareto-optimal solutions based on their solution approaches for the problem. Illustration of a simple network of three stages in supply chain network is shown in <Figure 15>.

4.3 Combination of Manufacturing and Logistics Model

In contemporary manufacturing, meeting promised delivery dates requires production schedules that take into account elements such as transportation. Previously scheduling focused on improvements in facility workloads and achieving cost reductions through mass production. Traditional manufacturing scheduling systems consider only manufacturing constraints while transportation scheduling systems typically consider only total vehicle mileage. These traditional systems cannot provide schedules that accommodate customer demands for just-in-time delivery (Okamoto, 2007).

It is difficult to develop a monolithic manufacturing scheduling system that includes large number of elements. Very large scheduling systems are difficult to develop and are inflexible. Furthermore, changes in system specifications often occur during system design. Okamoto *et al.* (2005) proposed a method for constructing a manufacturing system using XML to exchange data among small subsystems including a scheduler based on a genetic algorithm (GA). This system responds even to a slight change in the constraint by adding a penalty to the adaptation value, which is performed by the genetic algorithm.

Moon *et al.* (2004a, 2004b) integrated process planning and production scheduling in an APS model that included the factor of transportation. Okamoto *et al.* (2006a), (2006b) expanded Moon's model by integrating the manufacturing process and transportation between plants and proposed a solution based on the genetic algorithm. Most of the studies concerning scheduling that consider both manufacturing and transportation (Lee and Chen, 2001), (Soukhal *et al.*, 2005) have established a number of resource and transportation restrictions to systematize the problem from the viewpoint of complexity. These approaches offer only specialized solutions.

This section describes an integrated manufacturing and logistics model that incorporates pickup and delivery. In comparison with previous studies (Okamoto *et al.*, 2006a, Okamoto *et al.*, 2006b, Zhang, 2006), this model accommodates a single vehicle that transports multiple materials, intermediate, and finished products. We also developed a scheduler using a multiobjective genetic algorithm that minimizes both makespan and vehicle mileage.

4.3.1 Mathematical Formulation

When manufacturing and transportation schedules are created separately, the results of one become the constraints of the other. For example, the starting and completion times of the production schedule become time-window constraints in the transportation scheduling. On the contrary, the transportation schedule defines the earliest start time and latest completion time of the manufacturing process. Some approaches can reduce the constraint violation by considering the scheduling of manufacturing and transportation separately. However, some of these find a solution by combining local optimum solutions, but not by finding a global optimum. In addition, some approaches arrive at a solution that does not satisfy all constraints.

Most manufacturing scheduling problems, such as the JSP, are problems relating to process sequencing. For some problems, such as fJSP, machine selection is an issue. Transportation problems, such as the *vehicle-routing problem* (VRP), relate to routing (round-order sequencing) and vehicle-assignment. However, both manufacturing and transportation problems deal with two issues; sequencing and selection. Global optimization is possible only by deciding the sequence through integration, since it mutually and largely influences the result.

In our model, pickup and delivery services are considered as operations. For a production process using resources in different locations, service time and transportation time are considered. We then create different schedules by integrating pickup and delivery as operations.

In this problem, we make the following assumptions :

A1 : All resources including machines and vehicles are available at the same time (t = 0) and all operations can be started at $t = 0$.

A2 : The predecessors of manufacturing processes are given for each order.

A3 : At any given time, a resource can only execute one operation. A machine can execute a process and a vehicle can execute a pickup/delivery service or provide transportation. It becomes available for other operations once the operation currently assigned to it is completed.

A4 : When a process is assigned a different machine (B) from the previous process (A), then pickup at (A), transportation from (A) to (B), and delivery at (B) must be scheduled.

This model views the problem as a multiobjective optimization problem, and minimizes makespan and vehicle mileage. Objectives are as follows :

$$\min\ c_{\mathrm{M}} = \max_{i,k}\left\{c_{ki}\right\} \tag{15}$$

$$\min\ T = \sum_{v=1}^{V}\sum_{m=0}^{N}\sum_{n=0}^{N}\sum_{g\in U}\sum_{h\in U} d_{mn}a_{gm}a_{hn}y_{ghv} \tag{16}$$

Equation 15 is the objective function that minimizes the makespan. Equation 16 is the objective function that minimizes total mileage of all the vehicles. <Figure 16> describes that the product pickup does not occur earlier than the end of the process, and the material delivery is completed before starting the process. The detail descriptions are shown in Gen *et al.* (2008).

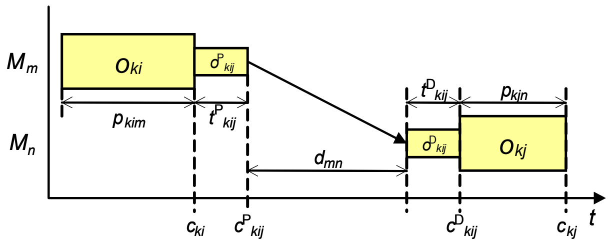

Figure 16. Operation time considering pickup and delivery

4.3.2 Multiobjective Hybrid Genetic Algorithm

The Multistage Operation-based Genetic Algorithm (Scheduler moGA) for the scheduling agent (Okamoto, 2007) is based on a multistage decision model. This algorithm uses an enhanced GA-based discrete dynamic programming (DDP) approach, proposed by Yang (Yang, 2001) for generating schedules in FMS environments. The scheduler moGA approach consists of two parts; sequencing operations and selecting resources. To apply the scheduler moGA to a new integrated scheduling problem, we modified a few chromosome designs.

To develop a multistage operation-based genetic representation for the problem, there are 3 main phases :

Phase 1 : Creating an operation sequence
- step 1.1 : Generate a random priority to each operation using encoding procedure for first vector v_1.
- step 1.2 : Decode a feasible operation sequence that satisfies the precedence constraints.

Phase 2 : Assigning operations to machine
- step 2.1 : Input the operations sequence found in step 1.2.
- step 2.2 : Generate a permutation encoding for machine assignment of each operation (second vector v_2).

Phase 3 : Designing a schedule
- step 3.1 : Create a schedule S using task sequence and processor assignments.
- step 3.2 : Draw a Gantt chart for this schedule.

In this section, the data set shown in <Table 6> is used for explanation.

Table 6. Simple example of processing time p_{kim}

	M_1	M_2	M_3
o_{11}	-	-	40
o_{12}	30	70	-
o_{13}	-	-	40
o_{21}	40	-	20
o_{22}	-	30	40
o_{23}	60	90	-

4.3.2.1 Sequencing Operations

A random key-based representation (Gonçalves *et al.*, 2005), (Gen, and Cheng, 2000) is used for the operation sequence. Any chromosome formed by crossover, mutation, and random generation is a feasible solution. It contributes to the reduction of the complexity of genetic operations, since the complicated repair process is not required. An example case of a chromosome is shown in <Figure 17>.

Operation ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Priority $v_1(l)$:	0.71	0.95	0.67	0.48	0.83	0.34	0.42	0.80	0.75	0.17	0.40	0.78	0.27	0.13

Figure 17. Chromosome v_1 drawn by random key-based encoding

A feasible operation sequence can be obtained through decoding. The result of this case is as follows :

$$\begin{aligned} Q &= \{ 4, 11, 1, 7, 12, 8, 5, 13, 14, 6, 2, 9, 10, 3 \} \\ &= \{ o_{21}, o^{P}_{212}, o_{11}, o^{P}_{112}, o^{D}_{212}, o^{D}_{112}, o_{22}, \\ &\quad o^{P}_{223}, o^{D}_{223}, o_{23}, o_{12}, o^{P}_{123}, o^{D}_{123}, o_{13} \} \end{aligned}$$

4.3.2.2 Selecting Resources

After completing the sequencing operations, the position of all stages (operations) is determined. Therefore, it is possible to select resources through a multistage decision making process. The delivery gene is not used in order to ensure that the same vehicle is always allocated for pickup. For this reason, in this example, genes of ID 8, 10, 12, and 14 are empty. As the result of this example, a chromosome can be drawn as shown in <Figure 18>.

Operation ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Resource $v_2(l)$:	3	1	3	3	2	2	4		4		4		4	

Figure 18. Chromosome v_2 drawn by resource permutation encoding procedure

A feasible operation sequence and resource selection can be obtained through decoding. The result of this case is as follows :

$$\begin{aligned} S = \{ &(o_{21}, M_3), (o^{P}_{212}, V_1), (o_{11}, M_3), (o^{P}_{112}, V_1), \\ &(o^{D}_{212}, V_1), (o^{D}_{112}, V_1), (o_{22}, M_2), (o^{P}_{223}, V_1), \\ &(o^{D}_{223}, V_1), (o_{23}, M_2), (o_{12}, M_1), (o^{P}_{123}, V_1), \\ &(o^{D}_{123}, V_1), (o_{13}, M_3) \} \end{aligned}$$

4.3.2.3 Scheduling

The schedule is created based on the decided operation sequence and resource selection. The starting time of each operation is allocated at the earliest possible time. It is best for the operation to end early for the two objective functions. But the service is deleted as there is no necessity for transportation in this step. In this example, o^{P}_{223} and o^{D}_{223} are deleted because o_{22} and o_{23} are assigned to the same machine. The scheduling result is as follows :

$$\begin{aligned} S = \{ &(o_{21}, M_3 : 0\text{-}20), (o^{P}_{212}, V_1 : 20\text{-}30), \\ &(o_{11}, M_3 : 35\text{-}75), (o^{P}_{112}, V_1 : 75\text{-}85), \\ &(o^{D}_{212}, V_1 : 40\text{-}50), (o^{D}_{112}, V_1 : 99\text{-}109), \\ &(o_{22}, M_2 : 50\text{-}80), (o^{P}_{223}, - : -), (o^{D}_{223}, - : -), \\ &(o_{23}, M_2 : 95\text{-}185), (o_{12}, M_1 : 109\text{-}139), \\ &(o^{P}_{123}, V_1 : 139\text{-}149), (o^{D}_{123}, V_1 : 163\text{-}173), \\ &(o_{13}, M_3 : 173\text{-}213) \} \end{aligned}$$

The Gantt chart in <Figure 19> shows this sample schedule.

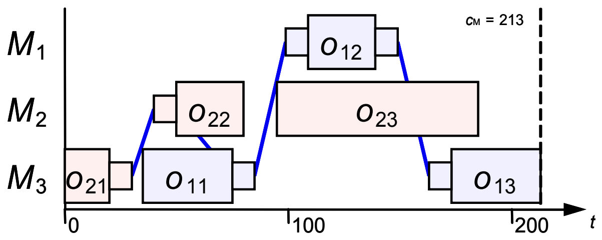

Figure 19. Gantt chart of scheduling result from the example dataset

4.3.2.4 Schedule improvement

Hybrid genetic algorithms (hGA) (Gen and Cheng, 2000) combine GA with other techniques, such as local search heuristics, in order to offset the problems inherent in GA. Changing operations in the critical path is an effective way of improving scheduling solutions. In this problem, we have two approaches to shift bottlenecks. One is to change the process order for each machine (Type A). The other is to change the service order in each vehicle (Type B).

In this study, a local search is applied only for the first rank, best-solution candidates. The solution candidates are divided into three areas, and different types of local searches apply to each area (<Figure 20>). Generally, in a scheduling problem, the decoding procedure consumes a great deal of time. Therefore, if local searches repeat calls to the decoding procedure for all chromosomes, the algorithm takes an enormous amount of time.

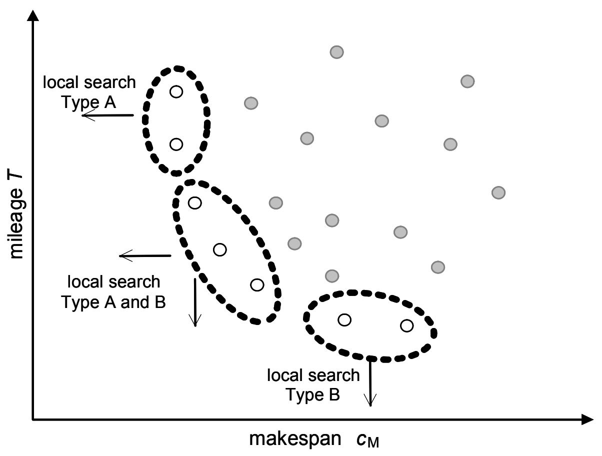

Figure 20. Local search for multiobjective problem

4.3.3 Numerical Experiment

The main module of this system is GA Scheduler as an implementation of algorithms that explained in previous section.

This example problem requires scheduling eight orders, in a six-machine, four-vehicle environment. The experimental dataset consists of processing timetable (<Table 7>) and machine locations (<Figure 21>), which are randomly generated. Other dataset and genetic parameters are :

$V = 1, 2, 3, 4$
$t^{U}_{kilj} = 30,\ \forall(k,\ i),\ (l,\ j)$
$p_C = 0.7,$
$p_M = 0.2,$
$popSize = 100,$
$maxGen = 100$

Table 7. Experimental dataset of processing time p_{kim}

	M_1	M_2	M_3	M_4	M_5	M_6
o_{11}	50	-	-	-	100	90
o_{12}	-	80	-	90	-	-
o_{13}	-	-	60	-	40	50
o_{21}	-	90	-	-	-	-
o_{22}	50	-	-	70	100	90
o_{23}	-	50	60	-	-	70
o_{24}	90	-	-	70	-	-
o_{31}	-	70	-	-	-	-
o_{32}	60	-	80	-	100	-
o_{33}	-	60	-	100	-	70
o_{41}	60	-	90	70	80	-
o_{42}	-	-	100	-	-	90
o_{43}	-	100	-	90	-	70
o_{51}	70	-	90	-	100	-
o_{52}	-	-	-	80	-	-
o_{53}	80	70	40	-	70	-
o_{54}	-	-	-	50	-	60
o_{61}	-	-	60	-	-	100
o_{62}	90	90	-	70	60	-
o_{63}	90	100	110	-	-	-
o_{71}	-	-	70	-	100	-
o_{72}	-	90	-	90	100	-
o_{73}	90	-	80	-	-	100
o_{81}	-	90	-	40	-	100
o_{82}	70	-	90	-	100	-
o_{83}	-	-	-	60	70	-
o_{84}	30	70	-	-	-	-

Local search methods :

(1) Apply local search to all solution candidates

(2) Proposed method

The Gantt charts of the representative experimental results solved by the local search methods (1) and (2) are shown in <Figure 22> and <Figure 23>, respectively. All best compromised solutions are shown as a Pareto graph in <Figure 24>. The final results show no difference between local search methods (1) and (2). However, the proposed method (2) required about 50～70% of the computation time of method (1). Then, its effectiveness was confirmed.

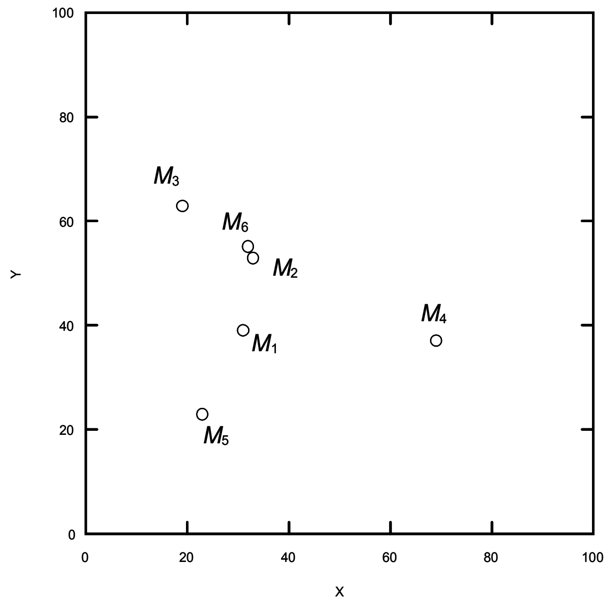

Figure 21. Resource locations for the numerical experiment

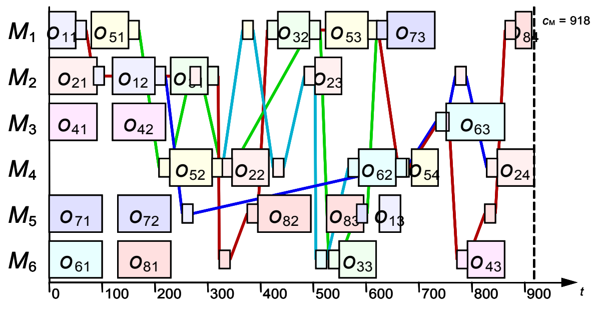

Figure 22. Gantt chart of an experimental result (V = 4, c_M = 918, T = 798)

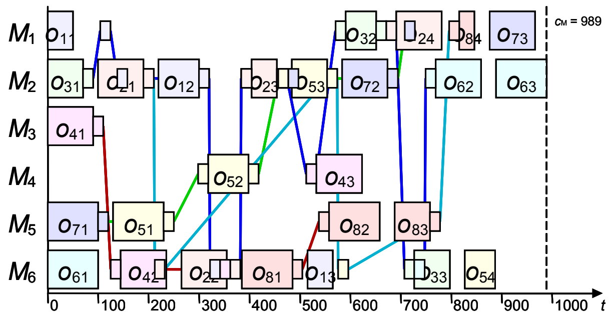

Figure 23. Gantt chart of an experimental result (V = 4, c_M = 989, T = 333)

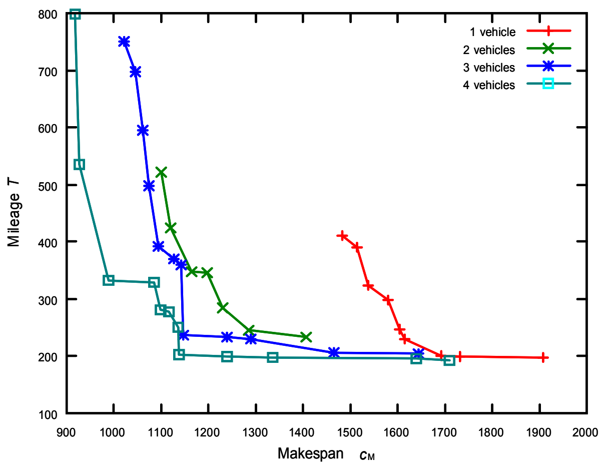

Figure 24. Best compromised solutions of the experimental results

5. Conclusions

In this survey paper we introduced the background of Advanced planning and scheduling (APS), and categorize the topics into three issues as many recent application of Evolutionary Technique are introduced based on some kinds of effective encoding methods.

Furthermore, for these issues we selected some hot topic and used some benchmark cases for making experiment : job-shop scheduling (JSP), assembly line balancing (ALB) model, and integrated scheduling models for manufacturing and logistics. For each topic, some effective encodings introduced to demonstrate concrete optimization problems among all the issues. Finally, we set out some related experimental result in case study and shown the performance of effectiveness of evolutionary approaches.

References

Adams, J., Balas, E., and Zawack, D. (1987), The shifting bottleneck procedure for job shop scheduling, *International*

Journal of Flexible Manufacturing Systems, **34**(3), 391-401.

Altiparmak, F., Gen, M., Lin, L. and Karaoglan, I. (2007), A steady-state genetic algorithm for multi-product supply chain network design, *Computers and Industrial Engineering*, Available online June.

Altiparmak, F., Gen, M., Lin, L., and Paksoy, T. (2006), A genetic algorithm approach for multiobjective optimization of supply chain networks, *Computers and Industrial Engineering*, **51**(1), 197-216.

Amiri, A. (2004), Designing a distribution network in a supply chain system : formulation and efficient solution procedure, *European Journal of Operational Research*, **171**(2), 567-576.

Anderson, E. J. and Ferris, M. C. (1994), Genetic algorithms for combinatorial optimization : the assembly line balancing problem, *ORSA Journal on Computing*, **6**, 161-173.

Baker, K. (1974), *Introduction to sequencing and scheduling*, New York, John Wiley and Sons.

Balas, E. (1969), Machine Scheduling via disjunctive graphs : An implicit enumeration algorithm, *Operation Research*, **17**, 941-957.

Bautista, J. and Pereira, J. (2002), Ant algorithms for assembly line balancing, *Lecture Notes in Computer Science*, **2463**, 65-75.

Bautista, J., Suarez, R., Mateo, M., and Companys, R. (2000), Local search heuristics for the assembly line balancing problem with incompatibilities between tasks, *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2404- 2409.

Baybars, I. (1986), A survey of exact algorithms for the simple assembly line balancing problem, *Management Science*, **32**, 909-932.

Bean, J. C. (1994), Genetics and random keys for sequencing and optimization, *ORSA Journal on Computing*, **6**, 154-160.

Becker, C. and Scholl, A. (2006), A survey on problems and methods in generalized assembly line balancing, *European Journal of Operational Research*, **168**, 694-715.

Blazewicz, J., Ecker, K. H., Schmidt, G., and Weglarz, J. (1994), *Scheduling in Computer and Manufacturing Systems*, Springer.

Bowman, E. H. (1960), Assembly line balancing by linear programming. *Operations Research*, **8**(3), 385-389.

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

Brucker, P. (1998), *Scheduling Algorithms*, Springer.

Brudaru, O. and Valmar, B. (2004), Genetic algorithm with embryonic chromosomes for assembly line balancing with fuzzy processing times, *The 8th International Research/Expert Conference Trends in the Development of Machinery and Associated Technology*, TMT 2004, Neum, Bosnia and Herzegovina.

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

Bukchin, J., and Tzur, M. (2000), Design of flexible assembly line to minimize equipment cost, *IIE Transactions*, 32, 585-598.

Chan, F. T. S. and Chung, S. H. (2004), A multi-criterion genetic algorithm for order distribution in a demand driven supply chain, *International Journal of Computer Integrated Manufacturing*, **17**(4), 339-351.

Chan, F. T. S., Chung, S. H., and Wadhwa, S. (2004), A hybrid genetic algorithm for production and distribution, *Omega*, **33**, 345-355.

Chen, C. and Lee, W. (2004), Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices, *Computers and Chemical Engineering*, **28**, 1131-1144.

Cheng, R., Gen, M., and Tsujimura, Y. (1996), A tutorial survey of job-shop scheduling problems using genetic algorithm, part I : representation, *Computers and Industrial Engineering*, **30**(4), 983-997.

Cheng, R., Gen, M., and Tsujimura, Y. (1999), A tutorial survey of job-shop scheduling problems using genetic algorithms, part II : hybrid genetic Search Strategies, *Computers and Industrial Engineering*, **36**(2), 343-364.

Dar-El, E. M. (1973), MALB-A heuristic technique for balancing large single-model assembly lines, *AIIE Transactions*, **5**(4), 343-356.

Dellaert, N., Jeunet, J., and Jornard, N. (2000), A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs, *International Journal of Production Economy*, **68**, 241-257.

Eck, M. (2003), *Advanced Planning and Scheduling*, BWI paper : http://obp.math.vu.nl/logistics/papers/vaneck.doc.

Erel, E. and Sarin, S. C. (1998), A survey of the assembly line balancing procedures, *Production Planning and Control*, **9**, 414-434.

Erol, I. and Ferrell Jr. W. G. (2004), A methodology to support decision making across the supply chain of an industrial distributor, *International Journal of Production Economics*, **89**, 119-129.

Falkenauer, E. and Delchambre, A. (1992), A genetic algorithm for bin packing and line balancing, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1189-1192.

Fisher, H. and Thompson, G. (1963), Probabilistic learning combinations of job-shop scheduling rules, *Industrial Scheduling*, **15**, 1225-1251.

French, S. (1982), *Sequencing and Scheduling. Mathematics and its applications*, Ellis Horwood Limited.

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 and Industrial Engineering*, **53**(1), 149-162.

Gao, J., Sun, L., and Gen, M. (2008), A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, *Computer and Operations Research*, **35**(9), 2892-2907.

Gen, M. and Cheng, R. (1997), *Genetic Algorithms and Engineering Design*, New York : John Wiley and Sons.

Gen, M. and Cheng, R. (2000), *Genetic Algorithms and Engineering Optimization*, New York : John Wiley and Sons.

Gen, M., Cheng, R., and Lin, L. (2008), *Network Models and Optimization : Multiobjective Genetic Algorithm Approach*, Springer, London.

Gen, M. and Syarif, A. (2005), Hybrid genetic algorithm for multi-time period production /distribution planning, *Compu-*

ters and Industrial Engineering, **48**(4), 799-809.

Gen, M., Altiparamk, F., and Lin, L. (2006), A genetic algorithm for two-stage transportation problem using priority-based encoding, *OR Spectrum*, **28**(3), 337-354.

Gen, M. and Zhang, H.(2006) : Effective designing chromosome for optimizing advanced planning and scheduling, C. H. Dagli *et al.*, eds. : *Intelligent Engineering Systems Through Artificial Neural Networks*, **16**, 61-66, ASME Press.

Ghosh, S. and Gagnon, R. J. (1989), A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems, *International Journal of Production Research*, **27**, 637-670.

Goncalves, J. F. and De Almedia, J. R. (2002), A hybrid genetic algorithm for assembly line balancing, *Journal of Heuristic*, **8**, 629-642.

Gonçalves, J. F., Magalhães Mendes, J. J., and Resende, M. G. C. (2005), A hybrid genetic algorithm for the job shop scheduling problem, *European Journal of Operational Research*, **167**(1), 77-95.

Grabot, B. and Geneste, L. (1994), Dispatching rules in scheduling : A fuzzy approach, *International Journal of Production Research*, **32**(4), 903-915.

Guillen, G., Mele, F. D., Bagajewicz, M. J., Espuna, A., and Puigjaner, L. (2005), Multiobjective supply chain design under uncertainty, *Chemical Engineering Science*, **60**, 1535-1553.

Held, M., Karp, R. M., and Shareshian, R. (1963), Assembly line balancing-Dynamic programming with precedence constraints, *Operations Research*, **11**, 442-459.

Helgeson, W. B., Salveson, M. E., and Smith, W. W. (1954), How to balance an assembly line, *Technical Report*, Carr Press, New Caraan, Conn.

Ida, k. and Osawa, A. (2005), Proposal of algorithm for shortening idle time on job-shop scheduling problem and its numerical experiments, *Journal of Japanese Industrial Management Assoc*, **56**(4), 294-301, (in Japanese).

Jackson, J. R. (1956), A Computing Procedure for a Line Balancing Problem, *Management Science*, **2**, 261-272.

Jain A. S. and Meeran S. (1998), A state-of -the-art review of job-shop scheduling techniques, technical report, department of applied physics, *Electronics and Mechanical Engineering*, *University of Dundee, Scotland*.

Jayaraman, V. and Pirkul, H. (2001), Planning and coordination of production and distribution facilities for multiple commodities, *European Journal of Operational Research*, **133**, 394-408.

Jayaraman, V. and Ross, A. (2003), A simulated annealing methodology to distribution network design and management, *European Journal of Operational Research*, **144**, 629-645.

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

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

Karp, R. M. (1972), Reducibility among combinatorial problems. In Miller R.E and Thatcher J.W. Editors : *Complexity of Computer Applications*, 85-104, New York : Plenum Press.

Khouja, M., Booth, D. E., Suh, M., and Mahaney, Jr. J. K. (2000), Statistical procedures for task assignment and robot selection in assembly cells, *International Journal of Computer Integrated manufacturing*, **13**, 95-106.

Kim, H. and Park, S. (1995), Strong cutting plane algorithm for the robotic assembly line balancing, *International Journal of Production Research*, **33**, 2311-2323.

Kim, Y. K., Kim, Y. J., and Kim, Y. H. (1996), Genetic algorithms for assembly line balancing with various objectives, *Computers and Industrial Engineering*, **30**(3), 397-409.

Kim, Y. K., Kim, Y., and Kim, Y. J. (2000), Two-sided assembly line balancing : a genetic algorithm approach, *Production Planning and Control*, **11**(1), 44-53.

Krasnogor, N. and Smith, J. (2000), A memetic algorithm with self-adaptive local search : TSP as a case study, *Proceedings of Genetic and Evolutionary Computation Conference*, July 10-12, Las Vegas, NV, 987-994, 2000.

Lee, C. Y. and Chen, Z.-L. (2001), Machine scheduling with transportation considerations, *Journal of Scheduling*, **4**, 3-24.

Leu, Y. Y., Matheson, L. A., and Rees, L. P. (1994), Assembly line balancing using genetic algorithms with heuristic generated initial populations and multiple criteria, *Decision Sciences*, **15**, 581-606.

Levitin, G., Rubinovitz, J., and Shnits, B. (2006), A genetic algorithm for robotic assembly line balancing, *European Journal of Operational Research*, **168**, 811-825.

Lopez, O. and Ramirez, M. (2005), A STEP-based manufacturing information system to share flexible manufacturing resources data, *Journal of Intelligent Manufacturing*, **16**(3), 287-301.

Mati, Y., Rezg, N., and Xie, X. (2001), An integrated greedy heuristic for a flexible job shop scheduling problem, *IEEE International Conference on Systems*, *Man*, *and Cybernetics*, **4**, 2534-2539.

MESX Joint Working Group. (2004), *MESX White Paper*, [Online]. Available : http://www.mstc.or.jp/faop/doc/informative/MESX-WP.pdf, (in Japanese).

Miltenburg, J. (2002), Balancing and sequencing mixed-model U-shaped production lines, *International Journal of Flexible Manufacturing Systems*, **14**, 119-151.

Moon, C. (2004b), *Evolutionary System Approach for Advanced Planning in Multi-Plant Chain*, Ph. D. dissertation, Waseda University, Japan.

Moon, C., Kim, J. S., and Gen, M. (2004a), Advanced planning and scheduling based on precedence and resource constraints for e-plant chains, *International Journal of Production Research*, **42**(15), 2941-2955.

Moscato, P. and Norman, M. (1992), A memetic approach for the traveling salesman problem : implementation of a computational ecology for combinatorial optimization on messagepassing systems, *Proceedings of the International Conference on Parallel Computing and Transputer Applications*, Amsterdam.

Najid, N. M., Dauzere-Peres, S., and Zaidat, A. (2002), A modified simulated annealing method for flexible job shop scheduling problem, *IEEE International Conference on Systems, Man and Cybernetics*, **5**(6).

Nicosia, G., Paccarelli, D., and Pacifici, A. (2002), Optimally balancing assembly lines with different workstations, *Discrete Applied Mathematics*, **118**, 99-113.

Noorul Haq, A., Jayaprakash, J., and Rengarajan, K. (2006), A hybrid genetic algorithm approach to mixed-model assembly line balancing, *International Journal of Advanced Manufacturing Technology*, **28**, 337-341.

Nowicki, E., and Smutnicki, C. (2005), An advanced tabu search algorithm for the job-shop problem, *Journal of Scheduling*, **8**(2), 145-159.

Okamoto, A. (2007), *Study on Integrated Scheduling for Manufacturing and Logistics Information System using Hybrid Genetic Algorithm*, Ph. D. dissertation, Waseda University, Japan.

Okamoto, A., Gen, M., and Sugawara, M. (2005), APS system based on scheduler moGA and XML, *Journal of the Society of Plant Engineers Japan*, **17**(2), 15-24, (in Japanese).

Okamoto, A., Gen, M., and Sugawara, M. (2005), APS System based on Scheduler moGA and XML, *Journal of the Society of Plant Engineers Japan*, **17**(2), 15-24, (in Japanese).

Okamoto, A., Gen, M., and Sugawara, M. (2006a), Integrated data structure and scheduling approach for manufacturing and transportation using hybrid multistage operation-based genetic algorithm, *Journal of Intelligent Manufacturing*, **17**, 411-421.

Okamoto, A., Gen, M., and Sugawara, M. (2006b), Integrated Scheduling Problem of Manufacturing and Transportation with Pickup and Delivery', *International Journal of Logistics and SCM Systems*, **1**, 19-27.

Orvosh, D. and Davis, L. (1994), Using a genetic algorithm to optimize problems with Feasibility constrains, *Proceedings of the First IEEE Conference on Evolutionary Computation*, 548-552.

Ozmehmet Tasan, S. and Tunali, S. (2008), A review of the current applications of genetic algorithms in assembly line balancing, *Journal of Intelligent Manufacturing*, **19**(1), 49-69.

PSLX Consortium (2003), *PSLX Technical Specifications, Recommendation*, Version 1.0, [Online]. Available : http://www.pslx.org/.

Raa, B. and Aghezzaf, E. H. (2005), A robust dynamic planning strategy for lot-sizing problems with stochastic demands, *Journal of Intelligent Manufacturing*, **16**(2), 207-213.

Rekiek, B. and Delchambre, A. (2006), *Assembly line design : The balancing of mixed-model hybrid assembly lines with genetic algorithms*, Springer Series in Advanced Manufacturing, London.

Rekiek, B., Dolgui, A., Delchambre, A., and Bratcu, A. (2002), State of art of optimization methods for assembly line design, *Annual Reviews in Control*, **26,** 163-174.

Roy, B. and Sussmann, B. (1964), Les problems d'ordonnancement avec contraintes disjonctives, Note D.S. No. 9 bis, *SEMA*, Paris, France.

Rubinovitz, J. and Bukchin, J. (1991), Design and balancing of robotic assembly lines, *Proceedings of the 4th World Conference on Robotics Research*, Pittsburgh, PA.

Rubinovitz, J. and Bukchin, J. (1993), RALB-a heuristic algorithm for design and balancing of robotic assembly line, *Annals of the CIRP*, **42**, 497-500.

Runarsson, T. P. and Jonsson, M. T. (1999), Genetic production systems for intelligent problem solving, *Journal of Intelligent Manufacturing*, **10**, 181-186.

Sabri, E. H. and Beamon, B. M. (2000), A multi-objective approach to simultaneous strategic and operational planning in supply chain design, *Omega*, **28**, 581-598.

Sabuncuoglu, I., Erel, E., and Tanyer, M. (2000), Assembly line balancing using genetic algorithms, *Journal of Intelligent Manufacturing*, **11**(3) 295-310.

Salveson, M. E. (1955), The assembly line balancing problem, *Journal of Industrial Engineering*, **6**, 18-25.

Scholl, A. (1993), *Data of Assembly Line Balancing Problems*, Schriften zur Quantitativen Betriebswirtschaftslehre 16/93, Th Darmstadt.

Scholl, A. (1999), *Balancing and Sequencing of Assembly Lines*, Physica-Verlag, Heidelberg.

Scholl, A. and Becker, C. (2006), State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, *European Journal of Operational Research*, **168**, 666-693.

Scholl, A. and Voss, S. (1996), Simple assembly line balancing-heuristic approaches, *Journal of Heuristics*, **2**, 217-244.

Soukhal, A., Oulamara, A., and Martineau, P. (2005), Complexity of flow shop scheduling problems with transportation constraints, *European Journal of Operational Research*, **161**, 32-41.

Su, P., Wu, N., and Yu, Z. (2003) Resource selection for distributed manufacturing in agile manufacturing, *Proc. of IEEE International Conference on Systems*, *Man and Cybernetics*, **2**(1), 1578-1582.

Suresh, G. and Sahu, S. (1994), Stochastic assembly line balancing using simulated annealing, *International Journal of Production Research*, **32**(8), 1801-1810.

Syam, S. S. (2002), A model and methodologies for the location problem with logistical components, *Computers and Operations Research*, **29**, 1173-1193.

Syarif, A., Yun, Y., and Gen, M. (2002), Study on multi-stage logistics chain network : a spanning tree-based genetic algorithm approach, *Computers and Industrial Engineering*, **43**, 299-314.

Talbot, F. B., Patterson, J. H., and Gehrlein, W. V. (1986), A comparative evaluation of heuristic line balancing techniques, *Management Science*, **32**, 430 - 454.

Tan, W. (2000), Integration of process planning and scheduling-a review, *Journal of Intelligent Manufacturing*, **11**(1), 51-63.

Tan, W. (2004), A linearized polynomial mixed integer programming model for the integration of process planning and scheduling, *Journal of Intelligent Manufacturing*, **15**(5), 593 -605.

Tavakkoli-Moghaddam, R., Jolai, F., Vaziri, F., Ahmed, P. K., and Azaron, A. (2005), A hybrid method for solving stochastic job shop scheduling problems, *Applied Mathematics and Computation*, **170**(1), 185-206.

Truong, T. H. and Azadivar, F. (2005), Optimal design methodologies for configuration of supply chains, *International*

Journal of Production Research, **43**(11), 2217-2236.

Tsai, D. M. and Yao, M. J (1993), A line-balanced-base capacity planning procedure for series type robotic assembly line, *International Journal of Production Research*, **31**, 1901 -1920.

W3C (2004), Extensible Markup Language (XML)

1.0 (3rd Edition), *W3C Recommendation*, [Online]. Available : http://www.w3.org/TR/2004/REC-xml-20040204.

Wu, Z. and Weng, M. X. (2005), Multiagent scheduling method with earliness and tardiness objectives in flexible job shops, *IEEE Trans. System*, *Man*, *and Cybernetics-Part B*, **35**(2), 293-301.

Xia, W. and Wu, Z. (2005), An effective hybrid optimization approach for muti-objective flexible job-shop scheduling problem, *Computers and Industrial Engineering*, **48**, 409-425.

Yan, H., Yu Z., and Cheng, T. C. E. (2003), A strategic model for supply chain design with logical constraints: formulation and solution, *Computers and Operations Research*, **30**(14), 2135-2155.

Yang, J. B. (2001), GA-based discrete dynamic programming approach for scheduling in FMS environments, *IEEE Transactions on Systems, Man and Cybernetics*, Part B, **31**(5), 824-835.

Yang, S. and Wang, D. (2000), Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling, *IEEE Trans. on Neural Networks*, **11**(2), 474-486.

Zhang, H. and Gen, M. (2005), Multistage-based genetic algorithm for flexible job-shop scheduling problem, *Journal of Complexity International*, **11**, 223-232.

Zhang, H. P. (2006), *Study on Evolutionary Scheduling Problems in Integrated Manufacturing System*, Ph. D. dissertation, Waseda University, Japan.