UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

A HYBRID GENETIC APPROACH TO SOLVE REAL MAKE-TO-ORDER JOB SHOP SCHEDULING PROBLEMS

Doctoral Thesis submitted at the Graduate Program in Production Engineering -UFSC

MARCO ANTÔNIO BARBOSA CÂNDIDO

Tampa, February 1997

Department of Industrial and Management Systems
College of Engineering
University of South Florida
4202 East Fowler Avenue, ENB 118
Tampa, Florida 33620-5350
(813) 974-2269

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

A HYBRID GENETIC APPROACH TO SOLVE REAL MAKE-TO-ORDER JOB SHOP SCHEDULING PROBLEMS

Marco Antônio Barbosa Cândido successfully defended his doctoral thesis on **April 10, 1997.** The committee approved his thesis for a **doctoral degree in Production Engineering.**

Paul E. Givens

Paul Givens, Ph.D.
Chair, Ind. & Mgmt. Sys. Engr., USF
Moderator

EXAMINING COMMITTEE:

Ricardo M. Bareia, Ph.D.
Chair, Grad. Prog. In Prod. Engr., UFSC
Supervisor

Suresh K. Khator

Suresh K. Khator, Ph.D.
Prof., Ind. & Mgmt. Sys. Engr., USF
Co-supervisor

Rogério Bastos, Dr. Eng.
Pro-rector, UFSC
Member

Fernando O. Gauthier, Dr. Eng.
Ass. Prof., Grad. Prog. in Prod. Engr., UFSC
Member

Michael Weng, Ph.D.
Ass. Prof., Ind. & Mgmt. Sys. Engr., USF
Member

To my father Antônio, my aunt Ita and my beloved wife Rísia

ACKNOWLEDGEMENTS

I am grateful to my supervisor, Prof. Ricardo M. Barcia, and my co-supervisor, Prof. Suresh K. Khator, for their precious guidance and support during my research. I would also like to thank all my committee members as well as the professors and secretaries from the Graduate Program in Production Engineering.

ABSTRACT

The existence of a gap between classical (theoretical) and real job shop scheduling problems has been reported by several authors

Local search procedures (e.g., tabu search and simulated annealing) and genetic algorithms have yielded very good results in classical job shop scheduling problems. However, these techniques present poor modeling abilities and very few applications including real production environment constraints have been reported. Moreover the search spaces considered in those few real world applications are usually incomplete and the real constraints included are small in number and environment dependent.

This work presents a robust hybrid genetic framework to solve job shop scheduling problems with large number of real world constraints, such as jobs with several subassembly levels, alternative processing plans for parts and alternative resources for operations, requirement of multiple resources to process an operation (e.g., machine, tools, fixtures, personnel), resource calendars, batch overlap, operation and job ready times, and sequence dependent setups. Also, the approach considers multiobjective evaluation functions.

The system uses modified schedule generation algorithms, which incorporate several decision support heuristics, to obtain a set of initial solutions. Each initial solution is enhanced by a local improvement procedure. Then a hybrid genetic algorithm, which incorporates a local hill climbing procedure, is applied to the set of local optimum schedules.

In order to support constraints and objectives of real production environments several modifications are proposed in the active and non-delay schedule generation algorithms, in the solution representation scheme and neighborhood structure of local search procedures, and in the genetic algorithm operators. Several tailored heuristics to be embedded in the basic algorithms are also proposed and analyzed.

The work is expected to reduce the gap between scheduling theory and practice, by enabling high performance scheduling techniques (constructive heuristics, local search procedures and genetic algorithms) to support real production environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1. SCHEDULING PROBLEMS

Scheduling is the allocation of resources over time in order to perform a set of operations and meet certain objectives while respecting a set of constraints. Scheduling problems appear in several areas. For instance, consider the scheduling of programs on computers, cars to be repaired in a garage, professors to classes in schools, physicians and nurses to patients in hospitals, production resources to jobs in manufacturing plants, etc.

In industrial scheduling problems a set of resources (e.g., machines, tools, personnel, fixtures) must be assigned over time to a set of operations in order to minimize some cost function. Industrial scheduling has been widely studied since the pioneering work of Johnson (1954) who proposed efficient algorithms for makespan minimization in problems of two and three stages. This thesis deals with the general make-to-order job shop scheduling problem which will be defined later.

The scheduling function is in the domain of production planning and control (PPC). The PPC can be decomposed in four interrelated levels:

1. ***Master production plan.*** The master production plan is generated based on demand forecasts (in make-to-stock production environments) and customer orders (in make-to-order manufacturers). The master plan tells the quantity and due date for each product.

2. ***Material and capacity requirements planning.*** A set of purchase and production orders must be elaborated to accomplish the master production plan. Also, an aggregate analysis of resource capacities and requirements is performed to guarantee the feasibility of the production plan.

3. ***Production Scheduling.*** Once the due dates and quantities for all products as well as the ready time for all orders, raw materials and components are

established, the resources must be assigned over time to perform operations in order to meet the production objectives (e.g., reduce work in process, minimize tardy jobs, etc.).

4. ***Shop control.*** A data collection and feedback system is used to control and monitor the execution of the production scheduling.

This work regards mainly to the third level, although the proposed system can also be used to aid the joint determination of material requirements, capacity planning, and production scheduling through what-if simulation.

Classifications for scheduling problems according to several criteria were reported by Graves (1981) and by Maccarthy and Liu (1993). Basically, scheduling problems are classified according to the following criteria:

1. ***Demand generation (make-to-order vs. make-to-stock):*** In make-to-order environments customers directly request the production orders. In make-to-stock environments production orders are generated based on demand forecasts and inventory replenishment policies, and scheduling also involves the determination of lot sizes.

2. ***Job processing and environment complexity***: The scheduling problem taxonomy of figure 1.1 covers the majority of classical scheduling problems:

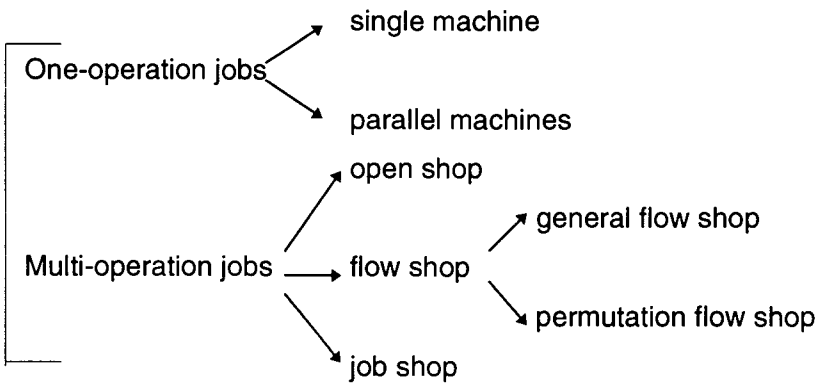


Figure 1.1. Scheduling Problem Taxonomy

One-operation jobs are jobs composed of only one operation. In one-operation, single machine problems all the operations must be executed on the same machine. In one-operation, parallel machines problems each operation can be processed on one of a set of identical or similar machines. A multi-operation job is composed of a set of operations, each operation processed on a different machine. In a flow shop all the multi-operation jobs have the same order of processing through the machines, i.e., all jobs have the same processing route. When a feasible solution to a flow shop scheduling problem requires all machines to process the same sequence of jobs, we have a permutation flow shop. The flow shop is only a particular case of the job shop problem, the most general scheduling problem. In a job shop each job can be processed by a different (but predefined) sequence of machines. An open shop scheduling problem occurs if the set of operations that composes a job can be processed in any sequence, i.e., when there is no predefined job processing route.

3. ***Performance measure***: A number of performance measures have been used to evaluate the scheduling quality. They can be classified as:

   - Criteria based on due dates: Maximum and mean tardiness, maximum and mean lateness, number of tardy jobs, etc. (Sen and Gupta, 1984).
   - Criteria based on resource utilization/flow time: Minimum and maximum completion time, minimum and maximum flow time, work in process level, etc.
   - Cost-based criteria: mean income loss, work in process cost, profit maximization, etc. (Benton, 1993).
   - Multi-criteria: analytical combination of single criteria or dominance analysis can be used to evaluate schedule quality according to more than one single criterion (Itoh, Huang and Enkawa, 1993; Daniels and Chambers, 1990; Sen and Gupta, 1984).

   A performance measure can also be classified as regular or non-regular. A regular performance measure can not be improved by delaying the completion of a job.

4. ***Dynamic nature of the production environment (dynamic vs. static environment):*** In a static environment the scheduling data (orders, resource availabilities, processing plans, etc.) are not allowed to change during the scheduling execution, i.e., new orders are not added and specifications are not altered during the scheduling execution. In a dynamic environment new orders can be added and specifications can change during the scheduling execution.

5. ***Data specification (deterministic vs. stochastic data):*** Problem data such as operation execution times and demand forecasts (in make-to-stock manufacturers) may be considered deterministic or stochastic.

Most scheduling problems are classified as NP-complete combinatorial search problems (Lenstra and Shmoys, 1995; Garey, 1979; King & Spachis, 1980; Goyal & Sriskandarajah, 1988), which means that the time required to find the solution exponentially increases with the problem size. There is no determinist algorithm able to solve a NP-complete problem in polynomial time. Hence heuristics able to find good solutions in reasonable amount of time are required. The computational complexity issue is addressed in the following section.

1.2. COMBINATORIAL SEARCH PROBLEMS

Solving a combinatorial search problem is to find one solution that satisfies a set of constraints among a very large but finite set of possible solutions (Karp, 1986). Well known examples of combinatorial problems are scheduling, routing, knapsack, assignment, VLSI circuit layout, Eulerian walk, Hamiltonian circuit, graph coloring, and satisfiability problems, among thousands of others.

Combinatorial search problems can be decision or optimization problems. Solving a decision problem requires only an answer to a yes or no question (e.g., is there any Hamiltonian circuit in a given graph?). Instead, solving an optimization combinatorial problem requires finding one arrangement among a set of feasible arrangements (e.g., determine the schedule that minimizes the makespan). Nevertheless, it is always possible by using simple tricks to transform an optimization problem in a decision

problem. This allows all combinatorial problems to be treated as decision problems without loss of generality.

Some combinatorial problems are easy and very efficient algorithms are available to solve them (e.g., Eulerian walk problem), while others extremely hard and solving them can require huge and impracticable computation effort (e.g, most scheduling problems). A problem is called tractable if it can be solved in a number of steps bounded by a polynomial in the size of its input (Edmonds, 1965), and intractable otherwise. For instance, several sorting methods are usually bounded by n^2 steps, where n is the number of elements to be sorted. Therefore sorting problems are tractable. Other problems (like scheduling problems) are bounded by a^n, n!, etc., and therefore are intractable.

Informally, a problem is in class *P* if it is tractable. Precisely, P is the set of all decision problems solvable in polynomial time. A formal formulation of this and other concepts presented in this section is beyond the scope of this thesis. They can be found in the book by Garey and Johnson (1979).

A decision problem D1 is *reducible* to a decision problem D2 if and only if problem D1 can be transformed in D2 by a polynomial time function. Therefore if D2 is in P then D1 will also be in P.

NP (non-deterministic polynomial) is the class of decision problems that are checkable in polynomial time, i.e., given a solution, one can check if the solution is correct in polynomial time. Of course all problems in P are also in NP, that is, $P \subset NP$. The question if P=NP has not been answered yet. However, it is highly probable that they are not the same set, because if P=NP finding a solution and checking a solution would present the same difficulty and this is not an appealing idea.

A decision problem D is NP-complete if it lies in NP and if all problems in NP are reducible to D. It means that if a polynomial algorithm exists to solve a NP-complete problem, then all others will also be solved in polynomial time. However, we must not expect this to happen. Instead, heuristics that find good (but not guaranteed optimal) solutions must be developed to overcome this likely inherent intractability of NP-complete

problems. The NP-complete problems are the most difficult combinatorial problems. The absolute majority of the scheduling problems are NP-complete. The list of known NP-complete problems is very large, containing hundreds of problems (Garey and Johnson, 1979).

1.3. THEORETICAL AND REAL JOB SHOP SCHEDULING PROBLEMS

The existence of a gap between scheduling theory and practice has been reported by several authors (Maccarthy and Liu, 1993; Mckay, Safayeni and Buzacon, 1988; Graves, 1981; Candido et al, 1995). A large number of benchmark job shop scheduling problems have been proposed by several authors for comparison purposes (Fisher and Thompson, 1963; Taillard, 1993; Drummond, 1996; Adams, Balas and Zawack, 1988; Applegate and Cook, 1991). Most of these problems, as well as most of the scheduling research performed so far do not consider some important issues of the real job shop (e.g., they usually consider the requirement of only one resource per operation). These theoretical benchmark problems are here named *classical* job shop scheduling problems as opposed to *real* job shop scheduling problems which this thesis intends to solve.

In the classical job shop scheduling problem (JSSP), a set of n jobs is processed on m machines. Each job is composed of a set of operations that have to be processed in a prespecified order. The problem is to find the best sequence of operations on each machine in order to minimize the maximum completion time (makespan) or any other single regular performance measure (Baker, 1974) without violating the precedence constraints. This problem was proved to be NP-complete. The following assumptions are adopted when dealing with classical JSSP:

- All the machines and jobs are available at time zero.
- Each machine can process only one operation at a time.
- There is only one unit of each machine type.
- A machine is the unique resource required to process an operation.
- Buffer space for work in process (WIP) is unlimited.
- Operation preemption is not allowed, i.e., once initiated, an operation must be completed before another operation be started on the same machine.
- Batch overlap is not allowed, that is, each batch is treated as a single unit.

- The environment is static, i.e., new jobs are not inserted in the scheduling during its execution and no machine breakdowns occur.
- All problem data are deterministic.
- Each job corresponds to only one part, i.e., bill of materials and assembly situations are not allowed.

A large variety of constraints and relaxations must be added to classical JSSP in order to represent real production environments. The number and type of constraints depend on the environment under consideration. It is impractical to design an universal scheduling system that considers all the characteristics of every production environments. Nevertheless, some constraints and alternatives have a high frequency of occurrence in real job shop environments. A scheduling system which models these commonly encountered characteristics would be able to solve a large number of industrial job shop scheduling problems that rise in practice. We include the following characteristics in a relevant subset of real constraints and alternatives:

- Several sub-assembly levels, i.e., existence of a bill of material for each product (job) ✓
- Additional renewable resources requirements to perform an operation (e.g., tools, fixtures, manpower)
- Alternative processing plans for each job: technologically different operation sequences or/and flexible precedence constraints on an operation sequence. ✓
- Alternative machines with possible different setup and processing times for each operation ✓
- Machine, tool, fixture and staff calendar (e.g. preventive maintenance, staff training) ✓
- Ready times for raw material and purchased components ✓
- Sequence dependent setup times ✓
- Batch overlap (different operations of the same batch being performed simultaneously) ✓
- Batch splitting and grouping
- Limited buffer space associated to each machine/work cell
- Upper and lower bounds to the waiting time of some operations

- Existence of transportation times and constrained availability of material handling devices

A system that intends to deal with a large number of real job shops must consider the above constraints and also be able to deal with multicriteria evaluation functions and work under dynamic conditions.

1.4. THESIS GOAL AND STRUCTURE

As it will be shown in next chapters, knowledge based systems and dispatching rules embedded in simulation models are the approaches that better represent real environment constraints. However, their performance in terms of evaluation criteria are usually worse than the performance obtained by local search procedures (like simulated annealing and tabu search), genetic algorithms, and tailored heuristics (like the shifting bottleneck procedure). On the other hand, the latter algorithms have poor modeling abilities and very few applications including real production environment constraints have been reported. The search spaces considered in those few real world applications are usually incomplete and the real constraints included are small in number and environment dependent.

This thesis intends to develop a robust framework using local search and genetic algorithms, beyond constructive heuristics, to deal with real make-to-order job shop scheduling problems. A large number of real world constraints are considered in the proposed model, such as jobs with several subassembly levels, alternative processing plans for parts and alternative resources for operations, requirement of multiple resources to process an operation (e.g., machine, tools, fixtures, staff), job and operation ready times, resource calendars, batch overlap and sequence dependent setups. Also, the approach considers multiobjective evaluation functions. To achieve this goal, a number of modifications are proposed in the active and non-delay schedule generation algorithms, in the solution representation scheme and neighborhood structure of local search procedures, and in the genetic algorithm operators. Several tailored heuristics to be embedded in the basic algorithms are also proposed and analyzed.

Therefore, it is expected a reduction of the gap between scheduling theory and practice, by enabling high performance scheduling techniques to support real production environments.

The thesis is organized as follows. Chapter two reviews the literature in the make-to-order job shop scheduling problem. Chapter three briefly describes Local Search and Genetic Algorithms and how they have been used to solve scheduling problems. In chapter four the models developed in this thesis to solve real JSSP are presented. Simulation results related to the proposed system are reported in chapter five. Finally, conclusions and further research suggestions are addressed in chapter six.

CHAPTER 2
APPROACHES TO SOLVE JOB SHOP SCHEDULING PROBLEMS

2.1. INTRODUCTION

A large amount of work in JSSP has been reported over the past three decades using several approaches: optimal methods, dispatching rules, constraint-based intelligent systems, Lagrangian relaxation, neural networks, tailored heuristics, tree search techniques, Petri nets, control theory, inductive learning models, local search procedures and genetic algorithms. A study comparing some of these techniques was presented by Tsang (1995). A number of good surveys and books have also been published (Pinson, 1995; Baker, 1974; Maccarthy and Liu, 1993; Graves, 1981; King and Spachis, 1980; Rodammer and White, 1988; Sen and Gupta, 1984; Cândido et al, 1995; Parker, 1995; Sule, 1996; Blazewicz et al, 1994; Zweben and Fox, 1994; Mattfeld, 1996; French, 1982; Brucker, 1995). These surveys are very useful due to the extraordinary amount of research developed in this area. However, most of them are focused on specific approaches (e.g., operational research, knowledge-based systems, etc.).

In this chapter a brief review of the different approaches to solve JSSP is presented. Two major classes of techniques can be distinguished: optimal methods and approximation methods.

2.2. OPTIMAL APPROACHES

The optimal approaches usually utilized by operational research scientists are used only to solve small instances of classical job shop scheduling problems due to the inherent intractability of the problem. The most used techniques are branch and bound, mixed integer programming and dynamic programming (King and Spachis, 1980). The best results achieved so far are due to Applegate and Cook (1991); Brucker, Jurisch and Sievers (1994); and Carlier and Pinson (1989, 1991, 1994). In these attempts sophisticated branch and bound methods based on disjunctive graph formulation were used to minimize makespan in classical JSSP. They solved small benchmark problems (e.g., 10x10 job shop of Muth and Thompson) in a reasonable amount of time, but (as

expected) they failed to solve medium and large problems in a reasonable time. Basically the branch and bound methods differ with respect to the elimination rules, the branching schemes, the bounding mechanisms, and the calculation of lower bounds in the search tree.

Good lists of scheduling problems solved by optimal methods are found in Maccarthy and Liu (1993) and in Vaca (1995). Other interesting applications of optimal methods are described by Sarin, Ahn and Bishop (1988), Alidaee (1993), Schrape and Baker (1978), and Gascon and Leachman (1988).

2.3. APPROXIMATION APPROACHES

Approximation methods or heuristics attempt to find good (but not necessarily optimal) solutions in a reasonable computational time. In a wide sense, heuristics can be defined as algorithms whose convergence to an optimal or even feasible solution can not be guaranteed. A review of application areas for heuristics, morphologic classifications and methodologies to design them are explained by Müller-Merbach (1981).

Because JSSP is NP-complete, a great research effort has been applied over the last three decades in the development of good scheduling heuristics. Many of these non-optimal heuristic techniques can deal with large problems (intractable for optimal techniques) in reasonable computational time. Note that heuristics can be developed to solve general or specific (environment-dependent) scheduling problems (He, Yang and Deal, 1993; Han and Dejax, 1991; Hertz and Widmer, 1989).

The main non-optimal techniques used to solve job shop scheduling problems are presented next. Special attention will be given to the so called metaheuristics which include local search and genetic algorithms.

2.3.1. Dispatching Rules

Dispatching rules are probably the most popular method in the scheduling of complex real job shops, and are very often embedded in schedule generation algorithms (see chapter 3) and in commercial scheduling packages.

Dispatching rules determine the operation to be processed next on a given resource among a set of schedulable operations. When a non-delay schedule generation algorithm is being adopted, the set of schedulable operations corresponds to a queue of tasks on the machine input buffer. Excellent surveys about dispatching rules were published by Blackstone, Phillips and Hogg (1982), Baker (1974), and Bertrand (1983). Dispatching rules are easy to implement and are computationally inexpensive. They can be based on processing times (e.g., SPT - Shortest Processing Time), due dates (e.g., EDD - Earliest Due Date), costs (e.g., margin profit based rule), or any other characteristic of the production environment. Rules obtained by the combination of single rules have also been applied. Dispatching rules can be classified as local or global rules. Local rules look at each machine individually while global rules dictate which operation to be scheduled next based on the entire shop status.

The performance of a dispatching rule depends not only on the evaluation function being used but also on a number of interconnected factors such as due date assignment method, tightness of due dates, machine loading level or job arrival rate, truncation method applied in certain rules, and production environment type (Blackstone, Phillips and Hogg, 1982; Baker, 1984; Bertrand, 1983; Kaplan and Unal, 1993; Bertrand, 1983; Ahiro, Isoda and Awane, 1984; Kannan and Ghosh, 1993).

2.3.2. Knowledge Based Systems

Since the middle eighties many knowledge based systems using constraint propagation techniques have been developed (Atabakhsh, 1991; Zweben and Fox, 1994; Rodammer and White, 1988; Kanet, 1987). Well known examples of these "intelligent" scheduling systems are ISIS (Fox and Smith, 1984), OPIS (Smith, 1994), OPAL

(Bensana, 1988), SOJA (Le Pape, 1985), Micro-Boss (Sadeh, 1994), ILOG (Le Pape, 1995), IxTeT (Laborie and Ghallab, 1995), among others.

These systems are able to deal with real instances of scheduling problems, i.e., several of the modeling requirements of real production environments described in chapter 1 are taken into account by knowledge based systems. Nevertheless they present high development cost and time, and their performances often depend on the performance of a human specialist (which is likely of low quality in a combinatorial situation). Moreover, the applications already implemented are usually environment-dependent and therefore general conclusions are difficult to draw.

2.3.3. Neural Networks

Recently a few feedback neural networks have been used to solve scheduling problems (Gallone, Chapillet and Alexandre, 1995; Peterson and Söderberg, 1993; Lo and Bavarian, 1991). Classical JSSPs were solved with a neural network based on the Hoppfield model implemented by Satake, Morikawa and Nobuto (1993). Chang and Jeng (1995) proposed an interesting neural network model to solve large instances of job shop scheduling problems.

Feedforward neural networks have also been used in hybrid scheduling systems as the ones developed by Sim, Yeo and Lee (1994), and by Rabelo et al (1993).

2.3.4. Lagrangian Relaxation

Lagrangian relaxation is a technique suitable to find lower bounds in combinatorial minimization problems. Basically, we take the primary integer formulation of the problem and relax some constraints into the objective function by attaching Lagrange multipliers to them. The relaxed problem (called LLBP - Lagrangian Linear Bound Program) is solved to optimality and a Lagrangian heuristic is applied to convert the LLBP solution into a feasible one (Beasley, 1993).

Lagrangean relaxation is a promising tool to deal with JJSP. Good results have been obtained by Hoitomt, Luh and Pattipati (1993), Luh and Hoitomt (1993), Chang and Chien (1993), Czerwinski and Luh (1994), Chen and Hsia (1994), Dobson and Khosla (1995), and Chen, Chu and Proth (1995). Some of these works consider a few of the real world constraints described in chapter 1, such as products with bill of materials and requirement of multiple resources per operation.

Lagrangian relaxation could also be classified as one of the simplified optimal approaches described next.

2.3.5. Simplified Optimal Approaches

Such models usually interrupt the search for the optimal solution after a certain CPU time, returning the current upper bound (King and Spachis, 1980), or reduce the search space by including a set of additional constraints (e.g., Bestwick and Lockyer, 1979). Relaxing constraints, modifying coefficients and decomposing the problem (e.g., Ghosh and Gaimon, 1993; Serafini and Speranza, 1994; Raman and Talbot, 1993) are other tricks that simplify the original problem and allow the application of optimal approaches.

A high performance procedure, which can be viewed as a truncated branch and bound algorithm, is the well-known shifting bottleneck procedure (Adams, Balas and Zawack, 1988). The algorithm ingeniously divides the scheduling problem into a set of one machine optimization and reoptimization problems.

Obviously, reaching the optimal solution can not be guaranteed by these approaches.

2.3.6. Local Search and Genetic Algorithms (Metaheuristics)

A number of research efforts have shown the high performance of local search procedures and hybrid genetic algorithms for solving small and large job shop scheduling problems. These algorithms are also called metaheuristics.

The most commonly used metaheuristics for scheduling purposes are genetic algorithms (Goldberg, 1989), tabu search (Glover and Laguna, 1993) and simulated annealing (Kirkpatrick, Gelatt and Vecchi, 1983). The theoretical basis of these techniques are described in chapter three. Applications of other local search techniques to JSSPs (e.g., threshold acceptance) have also been reported (Dueck and Scheuer, 1990; Aarts et al, 1994; LIn, Haley and Sparks, 1995).

Genetic algorithms and feedback neural network have strong similarities with local search procedures and many authors consider them special cases of local search procedures. This section conducts a literature review with respect to the use of metaheuristics for JSSPs. Special attention is addressed to GA approaches.

When dealing with local search algorithms, one must define a representation scheme and a neighborhood structure. Moving operators provide moves from the current solution to a neighboring one. The neighborhood structure can be analyzed in its completeness and uniqueness. The completeness regards to the capability of representing all feasible active schedules and the uniqueness concerns to the unique correspondence between a schedule and its representation. A representation is complete if it can represent all active schedules and is unique if for each schedule corresponds only one representation. A representation is redundant if each schedule can have more than one representation. A move in the neighborhood is considered legal or illegal if it yields feasible or infeasible solutions respectively. The schedule builder (also called decoding procedure) is an important external element that performs the mapping from the representation to the schedule itself. Depending on the application, part of the search procedure is performed by the schedule builder. Classical schedule builders are the semi-active, active and non-delay ones. All these concepts are detailed in chapter three.

Concerning to the search space topology for classical JSSP, a good analysis was performed by Mattfeld and Bierwirth (1996). Studying the "fitness landscape" of hard and easy benchmark problems, the authors addressed that the landscapes are difficult to

search for local search algorithms because the local optima are widely spread. However, the smoothness of the neighborhood suggests the use of adaptive search.

2.3.6.1 Genetic Algorithm (GA)

A large number of applications of GA for JSSP have been published over the last decade. Most of these genetic algorithms were proposed to solve the classical JSSP or small variations of the classical problem (Yamada and Nakano, 1995; Kim and Lee, 1994; Fang, Ross and Corne, 1993; Aarts et al, 1994; Mattfeld, 1996; Croce, Tadei and Volta, 1995; Park and Park, 1995; Bierwirth, 1994; Biegel and Davern, 1990; Bierwirth, Mattfeld and Kopfer, 1996; Falkenauer and Bouffouix, 1991; Reeves, 1995; Whitley, 1991). They differ from one another mainly in the representation schemes, in the schedule builder ability and responsibility, in the genetic operators, in the hybridization level with other heuristics (commonly used to generate high quality initial solutions or to improve previous generated solutions), and in the performance measure adopted. Many of these works have designed experiments to compare different operator types and rates, and representational schemes.

Related to the representations, one can use direct or indirect representation. In the direct representation (Bruns, 1993) the chromosome is the schedule itself and complex crossover operators are required to create new solutions. In the indirect representation simple operators are allowed and the chromosome must be mapped to a feasible schedule through a schedule builder (decoding procedure). An encoding scheme must also be analyzed by its completeness and redundancy level. Indirect representations are the most commonly used encoding schemes. Some indirect representation schemes encountered in the literature are job permutation, permutation of jobs with repetition, permutation of operations, permutation of operations per machine, and job-ordered list per machine. They differ in redundancy level, completeness, and complexity of the recombination operators required (see chapter three).

There are other "non-standard" and powerful GA models for the JSSP. In a number of works the GA operates on a set of dispatching rules. In the system proposed

by Herrmann, Lee and Hinchman (1995) a chromosome is a list of dispatching rules, one for each machine. In attempts due to Dorndorf and Pesch (1995), and Chiu and Yih (1995) the size of the dispatching rule list (chromosome) is equal to the number of operations and each rule (gene) is responsible for one dispatching decision. Storer, Wu and Vaccari, (1992) suggested that the chromosome could be formed by a dispatching rule list, each of these rules would control the sequencing process for a time window or a specific number of operations. These representations permit the use of simple recombination operators and allow easier implementations of complex scheduling environments, as shown by Herrmann, Lee, and Hinchman (1995).

Other interesting GA models used for JSSPs are those involving random keys which encode solutions with random numbers (Bean, 1994). The search is performed over the random key space, making easy the application of simple recombination operators.

Dorndorf and Pesch (1995) used a GA to determine the selection of nodes in the enumeration tree of the shifting bottleneck procedure. The chromosome is a permutation of machines determining the sequence of single machine problems for the shifting bottleneck procedure.

Some researchers have used GA unsupervised machine learning models (Goldberg, 1989). They apply bidding systems (Aytug, Koehler and Snowdon, 1994) and reward propagation techniques (Holsapple et al, 1993) to solve the JSSP. Aytug, Koehler and Snowdon (1994) proposed an adaptative dispatcher for a work cell with different parallel machines. The dispatcher owns a knowledge base whose rules are updated over the time through a GA learning system. In the work due to Chiu and Yih (1995), a GA operating on dispatching rule lists generates solutions that are used as training examples for a learning algorithm that acquires scheduling knowledge and represent it in a binary decision tree.

In problems involving more complex production environments, the search space may be explored partially by the schedule builder. However some studies have shown the

superiority of approaches that integrate problem specific knowledge in the GA representation and recombination operators (Uckum, Bagachi and Kawamura, 1993; Bruns, 1993), avoiding the schedule builder to perform any significant part of the search.

Many papers that attempt to schedule more realistic environments are concerned to specific facilities (Biegel and Davern, 1990; Syswerda, 1991; Hamada et al, 1995; Gilkinson, Rabelo and Bush, 1995). For instance, Hamada et al (1995) used a hybrid GA with rule based reasoning to solve a scheduling problem with a number of preferences (soft constraints) in a steelmaking company. An expert system was used to generate initial solutions to the GA, each satisfying at least one preference. The fitness function took into account the makespan and the violation of the preferences.

Some research attempts that consider additional constraints and alternatives related to real production environments are presented next. Particularly, we are interested in works that treat the additional real constraints as elements influencing directly the GA search.

A few papers report the use of GA in environments with alternative processing plans and alternative machines. In Uckum, Bagachi, and Kawamura (1993) an order has alternative processing plans and operations can be processed on alternative machines. The GA searches the space of all job order permutations. Using recombination operators processing plans are assigned to job orders and machines are assigned to operations specified in the plans. Nevertheless, the representation is incomplete and redundant, since the schedule builder assigns all the operations of the first job, then all the operations of the second job and so forth. Bruns (1993) attempted to solve a similar JSSP with a GA that used direct representation. In this model the schedule builder was eliminated and the representation was the schedule itself. The drawback of the direct representation was the complexity of the recombination operators needed to produce a "good" set of feasible schedules. The operators used in Bruns's work were not able to generate all active schedules. Gilkinson, Rabelo and Bush (1995) considered alternative machines per work cell. Each gene of the chromosome represented a work cell. A gene was composed of a list of operations and respective machines. The representation was complete and unique only because the production line is a flow shop. Also, some

operations were permitted to be simultaneously processed on the same machine. Holsapple et al (1993) used a hybrid GA/filtered beam search approach to schedule a flexible manufacturing cell where alternative machines and inter-station transfer time were allowed. The GA determined the job sequence (hence the representation was incomplete) and the filtered beam search chose the machines to process each operation of a predefined job sequence. The GA was implemented with an unsupervised learning procedure using reward propagation. Another GA using direct representation was implemented by Blume (1994). Alternative processing plans for each job and multicriteria cost functions were considered. Apparently, the GA searched for good sets of processing plans, while priority rules assigned operations to machines.

Sequence dependent setup times were considered by schedule builders in the GA implementations due to Rubin and Ragatz (1995), Gilkinson, Rabelo and Bush (1995), Syswerda (1991), and Uckum, Bagachi, and Kawamura (1993). Multiobjective criteria have also been incorporated into GA models (Uckum, Bagachi, and Kawamura, 1993; Syswerda, 1991; Hamada et al, 1995; Gilkinson, Rabelo and Bush, 1995; Blume, 1994). The dynamic nature of the problem was also studied by a number of researchers (e.g., Bierwirth et al, 1995). Lee and Shaw (1993) proposed a GA model to jointly solve the lot sizing problem and the scheduling problem in a flow shop environment. Gonzalez, Torres and Moreno (1995) reported a genetic algorithm for the no-wait flowshop problem, which is similar to a TSP. A complex scheduling problem with resource constraints was addressed by Syswerda (1991). The problem also considered task priorities, alternative setups and other weak constraints. Nevertheless, the GA manipulated only the task sequences, being the schedule builder responsible to build a legal schedule. The performance measure was based on the violation of weak constraints and on satisfaction of priorities.

In this section we reviewed the utilization of GA for scheduling problems. Most applications attempt to solve classical scheduling problems. Only a few works considered additional real world constraints. However, in these efforts the search spaces were incomplete and the real constraints included were usually small in number and environment dependent.

2.3.6.2. Tabu Search (TS)

The Tabu Search procedures have widely been applied to solve classical JSSPs (Barnes and Chambers, 1995; Dell'Amico and Trubian, 1993; Sun, Batta and Lin, 1995; Widmer, 1991; Barnes and Laguna, 1993). The applications are based on the disjunctive graph representation (see chapter three). Usually the neighborhood moves are obtained by reversing critical path arcs or making other changes in precedence relations on the longest path. Broader neighborhood structures also based on changing precedence relations on the critical path were proposed by Dell'Amico and Trubian (1993). Sun, Batta and Lin (1995) restricted the search to the active schedule space and the moves were performed by active chain manipulation. Tabu search applications for JSSP usually consider the makespan criterion. An exception is found in Widmer (1991), who used a TS approach to minimize a multiobjective function in the scheduling of a flexible manufacturing system with tool magazine capacity constraints. The main differences among the several TS applications for JSSPs are related to the search strategies (e.g., memory functions, aspiration criteria, etc.).

Compared to other local search procedures, Tabu Search approaches have yielded very good results in the classical JSSP making use of relatively low computational time.

2.3.6.3. Simulated Annealing (SA)

Simulated annealing is another powerful local search technique suitable to deal with classical JSSPs (Laarhoven, Aarts and Lenstra, 1992; Lin, Haley and Sparks, 1995; Aarts et al, 1994; Krishna, Ganeshan and Ram, 1995; Musser, Dhingra and Blankenship, 1993; Jeffcoat and Bulfin, 1993; Lourenco, 1995).

Most applications try to minimize the makespan. They adopt the disjunctive graph representation and the moves are based on reversing critical arcs or changing some other precedence relation on the critical path. Jeffcoat and Bulfin (1993) considered a resource constrained scheduling problem in parallel processors. The objective function to

be minimized was related to resource constraint violations, that is, this procedure tried to minimize the infeasibility. Simulated annealing algorithms usually require high computational time to achieve good solutions.

2.3.6.4. Other Local Search Procedures and Hybrid Models

An interesting local search procedure used to solve JSSP is threshold accepting (Dueck and Scheuer, 1990; Aaarts, Laarhoven, Lenstra and Ulder, 1994). It can be viewed as a deterministic simulated annealing, where a move is accepted if the difference in the evaluation function value between neighboring solutions is less than a non-negative threshold value. This value decreases during the search process.

Other important local search algorithms are multi-start iterative improvement (Aaarts, Laarhoven, Lenstra and Ulder, 1994), Twofold Look-ahead Search (Itoh, Huang and Enkawa, 1993), path algorithm (Werner, 1993), ant system (Dorigo Manniezo and Colorni, 1996), beam search, etc.

Pure genetic algorithms have performed worse than Tabu Search and Simulated Annealing for classical JSSP. This gap can be narrowed by hybridizing the GA with local hill climbing procedures, as shown by Matfeld (1996) and by Yamada and Nakano (1995). The hybridization allows the GA to work in the local optimum domain. Hybrid approaches between SA and GA (Lin and Hsu, 1993; Shen, Pao and Yip, 1994), and between TS and GA (Glover, Kelly and Laguna, 1995) have also shown to be promising since they incorporate advantages of one technique into another. Lourenço (1995) performed a computational study of local search algorithms hybrid with large-step methods for JSSPs. Sophisticated large-step moves were performed to diversify the search and local search procedures (as SA) operated after each large-step move.

2.3.7. Other Non-Optimal Approaches

The production scheduling can be modeled as a control system, whose inputs are job orders, material and resource requirements, disturbances (e.g., machine breakdown), and controlled inputs (e.g., maintenance schedule). The system state is described by

variables like WIP, resource availability, etc. The outputs can be for instance the inventory level of final products at time t (Rodamer and White, 1988).

Petri net theory has also been successfully used to model scheduling problems, as described by Lee and Jung (1995), by Xiong, Zhou and Manikopoulos (1995), and by Richard, Jacquet, Cavalier and Proust (1995).

Near-optimal (and a few optimal) techniques have been applied to resource constrained scheduling problems, as described by Blazewicz and Finke (1994). In inductive learning-based scheduling models (Piramuthu, Raman and Shaw, 1994) appropriate dispatching rules are selected to be applied according to the current state of the manufacturing system. There are also many other tailored heuristics used to solve JSSP. The review of all these methods is beyond the scope of this work.

2.4. CONCLUSION

Local search and hybrid genetic approaches have presented very good results in classical JSSP with makespan as criterion. However, the application of these procedures to real production environments is restricted to some specific facilities. Dispatching rules and knowledge based systems are still the most commonly used approaches to real world scheduling problems. Thus, the development of local search and genetic based scheduling systems that incorporate broad modeling capabilities is a promising research area.

CHAPTER 3
LOCAL SEARCH AND GENETIC ALGORITHMS

3.1. INTRODUCTION

In chapter 2 a literature review on the use of local search and genetic algorithms for job shop scheduling was presented. In this chapter these algorithms will be described with the purpose of explaining how they have been used to deal with *classical* JSSP. So this chapter provides the theoretical basis required to comprehend the scheduling models proposed in chapter 4 for *real* JSSP.

Both local search and genetic algorithms improve existing solutions. Therefore these algorithms require a constructive heuristic to generate initial solutions. Typical schedule generation algorithms are the semi-active, active and non-delay schedule generation algorithms. These classical procedures are described in next section. Afterwards, local search and genetic algorithms are addressed.

3.2. SCHEDULE GENERATION ALGORITHMS FOR CLASSICAL JSSP

In a semi-active schedule no operation can start earlier without changing a machine processing sequence or violating a technological constraint. An active schedule is a schedule where no operation can start earlier without delaying any other operation, i.e., an operation never waits on a machine input buffer if it can be completed before the next operation to be processed on the same machine arrives in the input buffer (Baker, 1974). The set of active schedules is a subset of the semi-active schedule set. All optimal solutions related to any regular measure are active. Therefore not all semi-active schedules need to be examined to find the optimal solution. Note that a regular performance measure is nondecreasing in job finishing time, i.e., it can not be improved by delaying the completion time of any job. The active schedule generation algorithm due to Thompson and Giffler (1960) for classical JSSP is presented below. Let:

PS_i = partial schedule at stage i, corresponding to the set of operations already scheduled at stage i.

S_i = set of schedulable operations at stage i, each of them corresponding to an operation whose preceding operation on the job has already been scheduled (inserted in PS_i).

The active schedule generation algorithm works as follows:

1) $i = 0$. $PS_i = S_i = \varnothing$.
2) Insert in PS_i the first operation of each job.
3) Determine the minimum operation completion time $\phi^* = \min_{u \in Si} \{l_u + t_u\}$, where l_u and t_u are the earliest start time and total processing time of operation u respectively. Let m^* be the machine required to complete operation u^* at time ϕ^*.
4) Determine the set $S_i' \subseteq S_i$ such that $S_i' = \{u / u \in S_i, l_u \leq \phi^*$, and u is processed on $m^*\}$.
5) Randomly select an operation $u^{(1)}$ from S_i' to be scheduled next.
6) Form PS_{i+1} by adding $u^{(1)}$ to PS_i. Form S_{i+1} by removing $u^{(1)}$ from S_i. Insert in S_{i+1} the operation that directly succeeds $u^{(1)}$ on the job.
7) $i = i + 1$.
8) If $S_i \neq \varnothing$ return to step 3; else stop and calculate the evaluation function value.

The above algorithm is able to generate all active solutions for a classical job shop scheduling problem, as proved by Thompson and Giffler (1960).

In a non-delay schedule no machine is kept idle if it could begin processing an operation, i.e., a machine is never idle if its input buffer is not empty. The non-delay schedule set is a subset of the active schedule set. An optimal schedule is not necessarily a non-delay one. The non-delay schedule generation algorithm for classical JSSP can be obtained by changing steps 3 and 4 of the procedure described above.

The non-delay schedule generation algorithm is presented below:

1) $i = 0$. $PS_i = S_i = \varnothing$.
2) Insert in PS_i the first operation of each job.
3) Determine the minimum operation start time $\phi^* = \min_{u \in Si} \{l_u\}$, where l_u is the operation u earliest start time. Let m^* be the machine required to start processing operation u^* at time ϕ^*.

4) Determine the set $S_i' \subseteq S_i$ such that $S_i' = \{u / u \in S_i, I_u = \phi^*$, and u is processed on $m^*\}$.
5) Randomly select an operation $u^{(1)}$ from S_i' to be scheduled next.
6) Form PS_{i+1} by adding $u^{(1)}$ to PS_i. Form S_{i+1} by removing $u^{(1)}$ from S_i. Insert in S_{i+1} the operation which is the direct successor of $u^{(1)}$ on the job.
7) $i = i + 1$.
8) If $S_i \neq \varnothing$ return to step 3; else stop and calculate the evaluation function value.

3.3. LOCAL SEARCH ALGORITHMS

Local search techniques have been successfully applied to classical JSSPs. The results reached by some tabu search and simulated annealing algorithms are among the best known for a number of benchmark job shop scheduling problems.

Local search techniques try to continuously improve solutions initially obtained by constructive heuristics. Given a solution s, the *neighborhood* of s is a set of solutions that can be derived by applying predefined slight modifications to s, i.e., by performing a *move*. So one must define a neighborhood structure and the related moving operators. The moving operators provide moves from one solution to another in the neighborhood. The *control strategy* determines whether the current solution will be replaced by a neighboring solution. The process continues until a termination criterion is fulfilled. For instance, consider a local hill climbing using the steepest descendent strategy. In this simple local search procedure the current solution (initially obtained by a constructive heuristic) is replaced by the neighboring solution that results in the greatest improvement in the evaluation function to be optimized. The process continues until a solution with no improving neighbor has been reached, i.e., until a local optimum has been found.

The basic elements of any local search procedure are the representation scheme, the neighborhood structure and the control strategy. Representation schemes and neighborhood structures are problem-specific. For example, a job shop scheduling problem and a graph coloring problem will use different representation schemes and neighborhood structures. In the following subsections the representation schemes and neighborhood structures more commonly used to solve classical JSSP will be presented. Then some basic local search strategies will be discussed.

3.3.1. A Representation Scheme for Classical JSSP

Almost all local search approaches to classical JSSP use the disjunctive graph representation scheme due to Roy and Susman (1964). In this formulation a job shop problem is represented by a graph G = (V, A ∪ H). The vertex set V corresponds to the set of operations, the arc set A connects consecutive operations of the same job, and the set of edges H consists of edges connecting operations processed on the same machine. When the edge set H is transformed into a conjunctive arc set S, i.e., when an orientation is given to all edges, and no cycle occurs, a solution is obtained. For practical purpose only the arcs belonging to the Hamiltonian path L_i of each machine i need to be represented in the solution digraph. Arc $(u,v) \in L_i \Leftrightarrow$ operation v is the operation processed after u on machine i. The final digraph obtained D = (V, A ∪ L), where $L = \cup L_i$, $L \subseteq S$, represents a particular schedule. The makespan corresponds to the length of the longest path in D.

For instance, suppose the rectangular classical JSSP of three jobs and three machines of table 3.1. The disjunctive graph G representing the problem instance and a digraph D of a particular solution are shown in figure 3.1, where vertex O_{ji} is the ith operation of job j. The dashed lines in graph G are the edges of H. In the solution graph D the edge set H was replaced by the Hamiltonian selection L. In this example the critical path corresponds to the vertex sequence (source, O_{31}, O_{11}, O_{12}, O_{21}, O_{32}, O_{33}, O_{13}, sink), and its length (makespan) is equal to 45 time units.

Job	Machine / processing time		
	first operation	second operation	third operation
1	M1 / 5	M2 / 7	M3 / 4
2	M2 / 10	M3 / 8	M1 / 2
3	M1 / 4	M2 / 12	M3 / 3

Table 3.1. Example - Problem Data

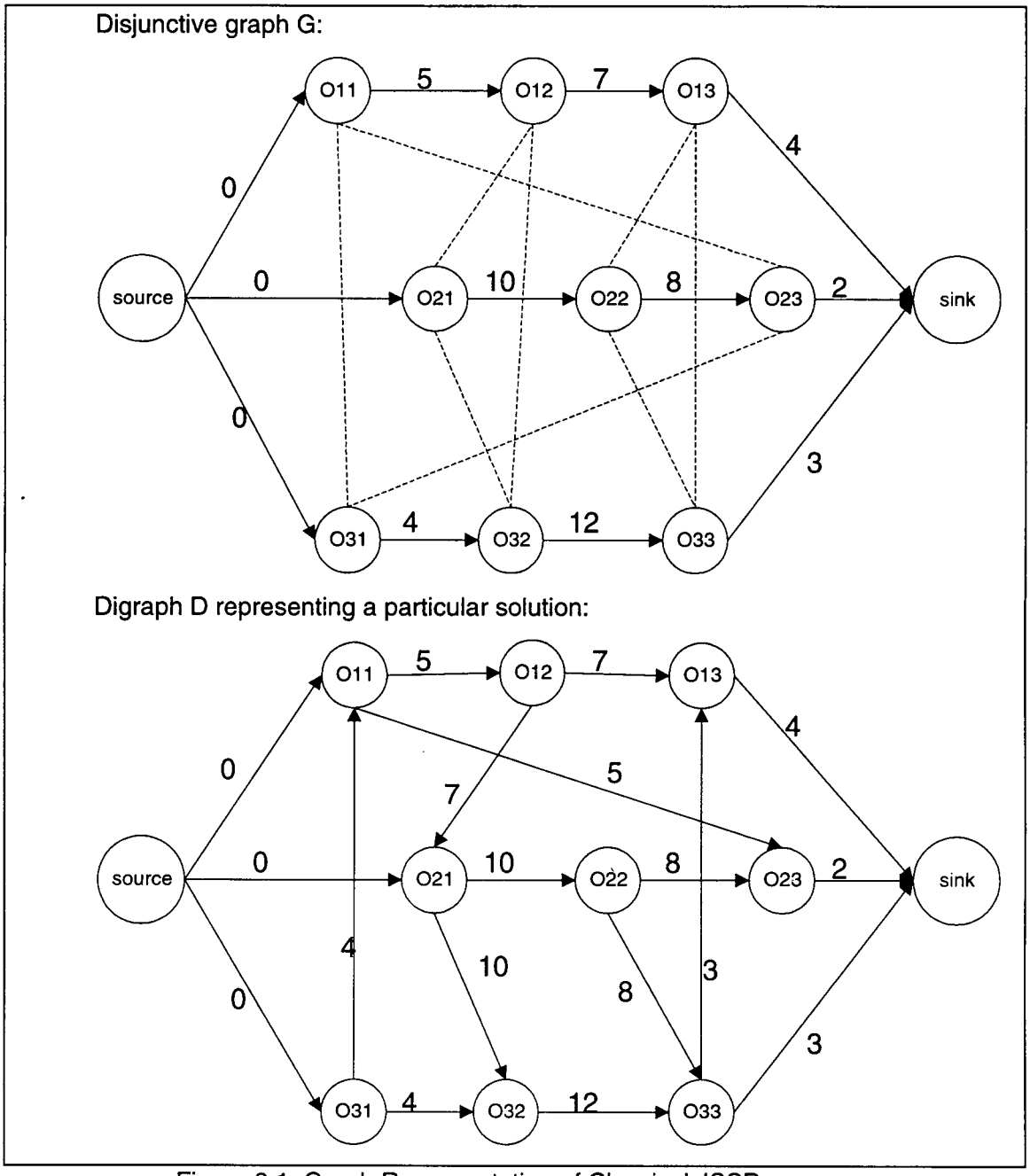


Figure 3.1. Graph Representation of Classical JSSP

3.3.2. Neighborhood Structures for Classical JSSPs

A neighborhood structure determines the set of neighboring solutions that can be reached from the current solution by performing a move or transition. Three important

properties of neighborhood structures are size, connectivity and feasibility (Mattfeld, 1996).

The neighborhood size, i.e., the average number of possible moves, is an important parameter to determine the computational time. A neighborhood structure holds the connectivity property if the optimal global solution can be reached from any solution through a finite number of moves (including non-improving moves). A move, characterized by a set of slight modifications in the current solution, can lead to a legal (feasible) or illegal (infeasible) solution. Neighborhoods that detain the connectivity property and whose moves always lead to feasible solutions are usually desirable. Nevertheless, when a local hill climbing algorithm is being implemented the connectivity property is not important anymore, because this local search strategy only allows improving moves. In this case it is desirable the smallest neighborhood structure for which all improving moves are available.

The absolute majority of local search applications in classical JSSPs adopt the disjunctive graph representation scheme, and the neighborhood moves are obtained by reversing arcs on the critical path or making other changes in precedence relations on the critical path. The following neighborhood structures presented good results when minimizing makespan in classical JSSPs:

N1 *(Laarhoven, Aarts and Lenstra, 1992): Given a solution digraph D, a move is generated by reversing an arc (v,w) on the critical path, such that operation w is the immediate successor of operation v on a machine k.* The reversal of (v,w) always results in a feasible (acyclic) solution. Moreover, the reversal of arcs on the critical path are the only arc reversals that can reduce the makespan value. N1 also holds the connectivity property. For instance, consider the solution represented by digraph D in figure 3.1 whose longest path is (source, O_{31}, O_{11}, O_{12}, O_{21}, O_{32}, O_{33}, O_{13}, sink). The neighborhood of D, N1(D), consists of the solutions obtained by reversing the following arcs: (O_{31}, O_{11}), (O_{12}, O_{21}), (O_{21}, O_{32}), and (O_{33}, O_{13}).

N2 *(Matsuo et al, 1988; Aarts et al, 1994): Given a solution digraph D, a move is generated by reversing arc (v,w) on the critical path, such that w is the immediate*

successor of v on machine k, and either the predecessor of v or the successor of w on machine k is not on the critical path, provided they exist. As in neighborhood N1, the reversal of (v,w) always produces a feasible solution. The reversal of arcs defined in N2 are the only arc reversals that can enhance a solution. However, as shown by Dell'Amico and Trubian (1993) N2 does not hold the connectivity property. Aarts et al (1994) showed that neighborhood N2 outperformed N1 for several benchmark JSSPs using different local search approaches. Obviously N2 is smaller than N1, since only a subset of moves of N1 is available in N2.

More intricate neighborhood structures were proposed by Dell'Amico and Trubian (1993) and Balas and Vazacopoulos (1994). These neighborhoods presented, however, larger sizes. As it will become clear in next chapter, large neighborhoods are not suitable for real JSSP. Since the objective here is to review the concepts on which our models are based, these more sophisticated neighborhood structures will not be described.

Note that neighborhood structures N1 and N2, as well as the more intricate neighborhoods proposed by Dell'Amico and Trubian (1993) and Balas and Vazacopoulos (1994) are proper only to minimize makespan, since the moves are based on reversal of arcs on the longest path.

3.3.3. Some Important Local Search Strategies

The simplest local search strategy is local hill climbing, where only improving moves are acceptable. As described by Reeves (1993) there are three basic hill climbing strategies. *Next descent* strategy selects the first found improving neighbor to replace the current solution. *Steepest descent* method searches the entire neighborhood and selects the neighbor which yields the greatest evaluation function improvement. *Random descent* randomly chooses an improving solution to replace the current solution. When there is no neighbor which enhances the evaluation function value the local hill climbing algorithm stops and the current local optimum is reported.

In order to extend the search, avoiding to finish the procedure at the first (and probably poor) local optimum found, more sophisticated strategies to control the search

process were developed. Among them Simulated Annealing (Kirkpatrick, 1983) and Tabu Search (Glover and Laguna, 1993) have yielded very good results for classical job shop scheduling problems. These algorithms attempt to find near-optimal solutions by allowing some moves toward worsening solutions.

Simulated annealing is an analogue of an algorithm used in statistical physics which simulates the cooling of a solid previously heated past its melting point back to its solid state. The properties of the cooled solid depend on the cooling rate. Simulated annealing can also be considered a special case of local search algorithm and it has been successfully utilized in combinatorial optimization. Simulated annealing is applied to classical JSSP as follows:

Let D be the current digraph solution and F(D) the makespan associated to D. The solution D is replaced by a randomly selected neighboring solution D' with probability P given by

$$P = \min \{1, \exp (- (F(D') - F(D))/c \}$$

where c is the control parameter corresponding to the temperature in the physic model. The value of c gradually decreases during the search process. At early iterations (high temperatures) the probability of accepting worsening moves is high. This probability decreases in the course of the annealing process, until the system freezes and no worsening solution is accepted. Note that improving solutions are accepted with probability 1 during the whole search. Using neighborhood structure N1 Laarhoven, Aarts and Lenstra (1992) proved that the annealing algorithm converges to a global minimum energy state (global optimum solution) if the sequence of values of c converges to zero, and the Markov chains generated at each temperature are of infinite length. This result is valid provided neighborhood structure N1 is adopted. Four parameters must be chosen to implement finite-time simulated annealing:

- the length of the Markov chains, i.e., the number of moves evaluated at each temperature
- the initial and final values of c (temperature)

- the decrement rule of c.

Tabu search is a powerful local search strategy to solve combinatorial search problems. Tabu search has presented marked success in classical job shop scheduling. In this approach the current solution is replaced by the best not forbidden neighboring solution (which is not necessarily better than the current solution). At each iteration there is a list of forbidden solutions. The forbidden solutions are a set of recently visited solutions. Actually, instead of a list of forbidding solutions, the algorithm keeps a list of forbidden moves that once performed will result in the forbidden solutions. Storing a list of forbidden moves, called tabu list (T), is computationally cheaper than storing a list of entire solutions. The existence of prohibited moves guides the search process to unexplored regions of the solution space. It also prevents cycling, i.e., it prevents that the algorithm moves from a local optimum solution D to a solution D' and then back to D, repeating this cycle indefinitely. Each time a solution D is replaced by a neighboring solution D', the move that would transform D' in D again is inserted at the end of the tabu list T, while the first move in T is removed, i.e., a move remains a tabu move during a certain number of iterations. In addition, an aspiration criterion is associated with each move. If the aspiration criterion is satisfied the move is considered admissible even it is a tabu move. Aspiration criterion is used for example to allow a tabu move that results in the best solution found so far. Other mechanisms like frequency-based memory are also used to coordinate intensification and diversification of the search process (Glover and Laguna, 1993).

Other local search approaches like threshold accepting (a deterministic version of simulated annealing) and multi-start iterative improvement have been used to solve JSSP. However, the results obtained were not as enthusiastic as the ones produced by tabu search, simulated annealing and genetic algorithms hybridized with local hill climbing.

3.4. GENETIC ALGORITHMS

Genetic algorithms have been widely used to solve hard combinatorial problems like job shop scheduling problems.

As described by Goldberg (1989) genetic algorithms are robust (efficient and efficacious) optimization methods abstracted from nature's adaptation process. GAs search from a population of strings (chromosomes) corresponding to encoded solutions. Individuals from the population are selected to reproduce based on survival of the fittest, i.e., the fitter the individual the higher the probability of being chosen to reproduction. Reproduction, crossover and mutation are applied to successive populations of individuals to generate new and likely fitter populations. Reproduction is the copy of chromosomes into successive generations according to their fitness, i.e., the higher the chromosome fitness the higher the probability of the chromosome being entirely or partially copied into the next generation. Crossover is a random exchange of sections of the parent's chromosomes. Together, reproduction and crossover conduct the adaptation process by reproducing and combining high quality genetic materials in successive generations. Mutation operators, which randomly perform slight modification in the chromosomes, are used in attempt to recover high-performance genetic materials that eventually are eliminated from the gene pool by reproduction and crossover.

In GA terminology *chromosomes* are strings representing solutions. *Genes* correspond to variables or features, the values of the variable or feature are called *alleles*. The position of a gene in a chromosome is called *locus*. The encoded structures (chromosomes) correspond to the *genotype*, while the solutions themselves obtained by decoding the chromosomes are the *phenotype*. The *fitness* of an individual or chromosome can be interpreted as the evaluation function value associated to the chromosome.

Several variations of genetic algorithms have been proposed. A basic GA template is presented below, where i is the generation counter:

1) i=0;
2) Generate an initial population of solutions P(i) using a random generation algorithm;
3) While the termination criterion is not satisfied do
    3.1) Generate P(i+1) from P(i) by applying reproduction, crossover and mutation;
    3.2) i = i+1;
4) Return the best individual in P(i);

Step 3.1 of the basic GA template is detailed below, where rand(1) is a randomly generated real number between 0 and 1:

a) Decode the M individuals of population P(i) and evaluate them, i.e., calculate their fitness values.
b) Initialize P(i+1) empty;
c) While the number of individuals in P(i+1) is less than M do
    c.1) Select a couple of parents p_1 and p_2 from P(i) with probability proportional to their fitness;
    c.2) If rand(1) < crossover rate then
        apply the crossover operator to p_1 and p_2 obtaining new individuals o_1 and o_2;
    else
        $o_1=p_1$ and $o_2=p_2$;
    c.3) For j=1 to 2 do
        If rand(1) < mutation rate then
            apply the mutation operator to o_j obtaining o_j';
        else
            $o_j'=o_j$;
    c.4) Insert o_1' and o_2' in P(i+1);

The formal mathematical validation of genetic algorithms can be simplified by introducing the concept of *schema* (similarity template). Considering only binary representations a schema is a string over the extended alphabet {0,1,*}, where * is a don't care symbol, i.e., * can be 0 or 1. For instance the two strings

10011010
01110110

are examples of the schema

***1**10

Short length, low order, high performance schemata (also called *building blocks*) are combined by the GA producing probably better individuals during the adaptation process. The incidence of these above average desirable schemata exponentially increases in successive generations.

Other important characteristic of a GA is its implicit parallelism. Although at each generation the computational effort performed is proportional to the population size n, about n^3 schemata are processed in parallel. A formal schemata analysis is reported by Goldberg (1989).

Binary encoding schemes and related traditional crossover operators like simple crossover, two points crossover, and uniform crossover are not suitable for most combinatorial problems, including job shop scheduling problems. For these problems higher-cardinality alphabets must be utilized. For example, permutation schemes are used in a number of combinatorial problems. Solutions to a traveling salesman problem (TSP) can be encoded by a permutation of letters, each representing a city. The sequence of letters in the chromosome corresponds to the sequence in which the cities are visited.

Combinatorial problems usually present high degree of *epistasis*. Epistasis occurs when changes in gene frequencies at different loci are not independent. Epistatic problems require non-standard crossover operators able to produce feasible offspring and maintain important blocks of dependent linked loci. Also, special mechanisms to increase selection pressure must be used in epistatic domains (Goldberg, 1989).

Next subsections briefly describe the coding schemes, genetic operators, and population management strategies extensively used in applications of GAs for classical job shop scheduling problems.

3.4.1. Coding Schemes for Classical JSSPs

Application of GAs in classical job shop scheduling problems are usually carried out by non-binary indirect representation schemes. We should remember that a representation scheme is complete if it can represent all optimal schedules and non-redundant if there is an one to one relationship between the genotype and the phenotype. Redundancy causes false competition. The most used representation schemes and their main characteristics are listed below:

1) Job Permutation (Uckum, Bagachi and Kawamura, 1993; Biegel and Davern, 1990; Hamada et al, 1995; Whitley, 1991):
    - Chromosome: permutation of jobs
    - Low redundancy level
    - Incomplete

2) Permutation with repetition (Fang, Ross and Corne, 1993; Bierwirth, Mattfeld and Kopfer, 1996; Bierwirth, 1994; Mattfeld, 1996):
    - Chromosome: permutation of jobs with repetition. Each job appears in a permutation as many times as its number of operations
    - Medium redundancy level
    - Complete (completeness depends on the decoding procedure)

3) Permutation of operations (Kim and Lee, 1994, 1995):
    - Chromosome: *preference* list of all operations
    - Highly redundant
    - Complete (completeness depends on the decoding procedure)

4) Permutation of operations per machine (Falkenauer and Bouffouix, 1991; Croce, Tadei and Volta, 1995):
    - One subchromosome per machine. Each subchromosome is a *preference* list of operations to be processed in the machine
    - Low redundancy level
    - Complete (completeness depends on the decoding procedure)

5) Disjunctive graph / job-ordered list per machine (Aarts et al, 1994; Park and Park, 1995, Yamada and Nakano, 1995):
    - One (sub)chromosome per machine. Each is a ordered list of operations to be processed in the machine
    - Unique (non-redundant)
    - Complete

Although the last representational scheme is complete and unique, it requires more complex recombination operators to produce feasible solutions.

An overview of crossover and mutation operators used for classical JSSPs is given in the following section.

3.4.2. Crossover and Mutation Operators for Classical JSSPs

Genetic operators are related to representation schemes. A survey of crossover operators for ordering applications was presented by Poon and Carter (1995). Considering representation schemes based on permutation without repetition (schemes 1, 3 and 4) the following crossover operators have been widely used:

- ***PMX (partially matched crossover):*** The two parental strings are aligned and two crossover points are chosen at random. The region between these points defines a matching section or interchange mapping in which exchanges between parents are performed point by point. PMX tends to keep the absolute positions of elements.

- ***LOX (linear order crossover):*** A matching section is determined in a similar way to PMX. At first the elements in parent 2 that occur in the matching section of parent 1 are deleted. Then the remaining part of parent 2 is combined with the matching section of parent 1 such that the exchanged substring keeps its original position. The other offspring is obtained in the same manner. LOX tends to respect relative positions.

- ***UX (uniform order crossover):*** The offspring chromosome is initialized empty. At each position one of the two parents is randomly chosen and the element at the first position in the chosen parent is inserted next in the offspring string. The element is then deleted from both parents. Complementary choices at each position determine the other offspring.

Examples of UX, PMX and LOX crossover operators are shown in figure 3.2. Some other useful crossovers for permutation-based representations are C1 (Reeves,1992), cycle crossover (Goldberg, 1989), and edge-recombination crossover (Whitley et al, 1991).

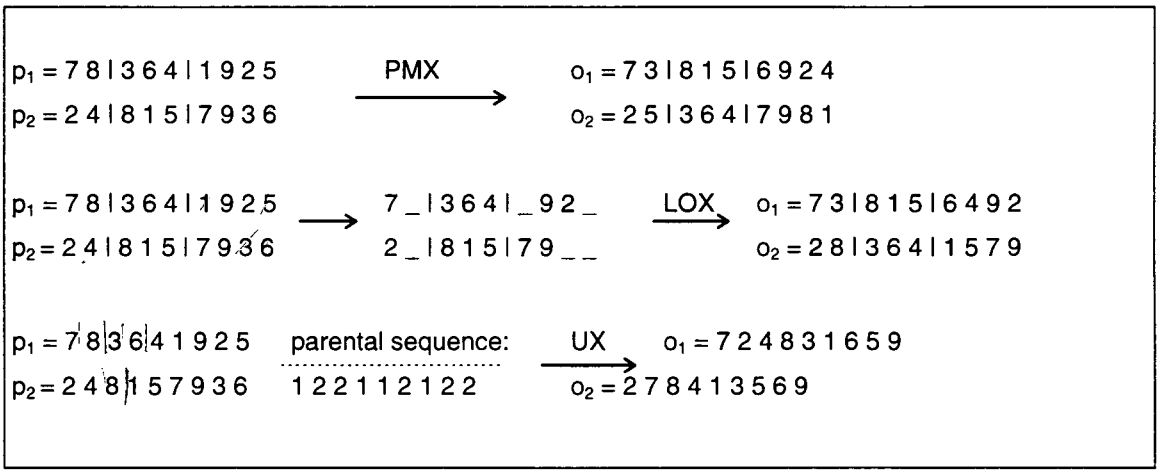


Figure 3.2. PMX, LOX and UX Crossover Operations

Some of these crossover operators were modified to deal with representations based on permutation with repetition. Bierwirth (1996) proposed the generalized order crossover (GOX) which is an extension of LOX for permutation with repetition. Similarly

Mattfeld (1996) proposed the generalized position crossover (GPX) and the generalized uniform crossover which are generalizations of PX and UX respectively.

Several mutation operators have been applied to permutation-based representations in job shop scheduling problems. *Swap mutation* exchanges the values (alleles) of randomly chosen adjacent loci. When the two loci (whose alleles are exchanged) are not necessarily adjacent, we have the *order mutation.* The *position mutation* randomly changes the position of an element (allele) in the chromosome.

In most applications crossover rates are between 0.5 and 1.0, and mutation rates are between 0.01 and 0.1.

3.4.3. Population Related Factors for Classical JSSPs

Seeding, selection method, fitness scaling, population size, termination criterion, decoding procedure, hybridization level, among others, are important factors in the performance of GAs in classical JSSPs. Here it will be described only commonly adopted variations of these factors. Further details on these issues are reported by Goldberg (1989) and Reeves (1993).

Seeding is the generation of the initial population. Semi-active, active and non-delay schedule generation algorithms are frequently used to generate an initial population of schedules. Other heuristics (e.g., dispatching rules) can be embedded in these procedures to improve the quality of initial solutions. A good set of initial solutions can help the GA to reach better solutions, but can also lead to a premature convergence.

In traditional GAs, parents are stochastically selected for reproduction through a hypothetical roulette wheel, i.e., the probability of a chromosome be selected is proportional to its fitness. This selection scheme is called *stochastic sampling with replacement.* Several other selection schemes were proposed (De Jong, 1975). *Deterministic sampling and expected value models* eliminate or reduce the variance in the roulette wheel selection, that is, they try to eliminate or reduce the difference between the expected and actual number of copies of an individual. Overlapping populations can be

implemented by selecting a proportion of the population for reproduction. Their offspring replace only part of the population members. This method is called *incremental replacement. Elitist selection* ensures the inclusion of the best individual of the current population into the next one. Baker (1985) proposed a *ranked-based selection* where individuals within the population are sorted according to their fitness and this ranking is used to guide the selection. In epistatic domains (including JSSPs) the use of increased selection pressure is required to avoid a tedious and almost non-improving search. Severe selection methods like elitist and ranked-based selections are more suitable for epistatic problems (Mattfeld, 1996).

Fitness scaling are also used to control the selection pressure. At early stages there exist a large number of poor solutions and only a few good solutions. If absolute fitness values are used in the selection process the GA will prematurely converge. On the other hand, at later stages the fitness variance within the population is small and the search process will be similar to a random walk if absolute fitness values are used in the calculation of selection probabilities. Scaling the fitness values at each generation overcomes these problems. Let f be the absolute fitness value of an individual. The scaled fitness f' is given by

$$f' = af + b$$

where parameters a and b can be determined by several ways. Goldberg (1989) suggests that a and b must be obtained from the conditions (considering maximization problems):

1. $f'_{average} = f_{average}$
2. $f'_{max} = 2f_{max}$ (if $f'_{min}<0$ then the condition $f'_{min} = 0$ replaces the condition $f'_{max} = 2f_{max}$)

where $f_{average}$, f_{min} and f_{max} are respectively the average, minimum and maximum fitness values within the population. A more severe selection (Mattfeld, 1996) can be achieved by determining the values of a and b such that:

1. $f'_{min} = 0$
2. $f'_{max} = f_{max} - f_{min}$

These transformations can be easily extended for minimization problems.

Population size is an important GA parameter. A small size drives the population to a restricted small portion of the solution space, while a large population size is computationally expensive. Although better results are reached by increasing the population size, Nakano (1994) observed a saturation of this trend. Also, ideal population size seems to be dependent on the chromosome length (Reeves, 1993).

Termination criteria are usually based on a fixed number of generations, or on a population convergence metric (like entropy). For classical JSSPs sophisticated convergence-based termination criteria can produce non-desirable results (Mattfeld, 1996), once significant improvements can be achieved after several generations of stagnation (phenomenon also observer in our GA implementation for real JSSPs). In other cases high genetic diversity can exist during several generations without any significant improvement be achieved.

The decoding procedure is used to obtain the solution phenotype (and associated fitness value) from the solution chromosome. Semi-active, active and non-delay schedule generation procedures have been used to decode strings of characters into schedules. The sequence of alleles in the chromosome determines, at each stage, the operation to be scheduled next from a set of schedulable operations.

As mentioned earlier the hybridization of the genetic algorithm with a local hill climbing was shown to enhance the GA performance in classical JSSPs. This can be achieved by applying a local hill climbing procedure to each individual right after it has been decoded.

Several of the GA concepts described here for classical JSSPs will be useful in the development of a scheduling system for real JSSPs.

CHAPTER 4
THE PROPOSED MODEL

4.1. SYSTEM SCOPE

As described in earlier chapters, local search approaches like Tabu Search (TS) and Simulated Annealing (SA) have yielded very good results in classical JSSP when minimizing makespan. These approaches use the disjunctive graph representation scheme with moves obtained by reversing arcs on the critical path. Such critical arcs reversing moves are suitable only to minimize makespan. Also, the disjunctive graph can not easily represent real life constraints, although some insights were presented by White and Rogers (1990).

We have also discussed that pure genetic algorithm (GA) methods usually perform worse than TS or SA for well known benchmark JSSP. This performance gap can be eliminated by hybridizing the genetic algorithm with a greedy local search procedure, as shown by Yamada and Nakano (1995) and Mattfeld (1996). These attempts were however limited to makespan minimization in classical job shop scheduling problems. Genetic-based scheduling systems that take into account some real production environment constraints have previously been implemented. However, the search space considered is usually incomplete and the real constraints included are small in number and are environment dependent.

This work attempts to develop a robust framework to deal with real job shop scheduling problems. In order to support constraints and objectives of real production environments a number of modifications are proposed in the active and non-delay schedule generation algorithms, in the solution representation scheme and neighborhood structure of local search procedures, and in the genetic algorithm operators. The following commonly encountered characteristics are considered in this framework:

1. *Precedence and resource capacity constraints.* These are the classical JSSP constraints. They assure that precedence relationships among operations of the same part are respected and that each resource processes at most one operation at a time.

2. *Several subassembly levels for each job.* A job (final product) can be composed of several parts. Precisely, a bill of material is defined for each job. For instance, consider the following bill of material of a job:

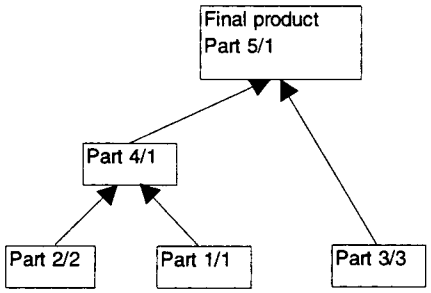


Figure 4.1. Bill of Material

In order to produce 1 unit of the job (final product) 1 unit of part 4 and 3 units of part 3 are required. Similarly producing 1 unit of part 4 requires 2 units of part 2 and 1 unit of part 1. Here part 4 is called the *succeeding* part of parts 1 and 2, as well as part 5 is the succeeding part of part 4 and 3. On the other hand parts 4 and 3 are the *preceding* parts of part 5, and part 1 and 2 are the preceding parts of part 4. Parts 1, 2 and 3 are at the first level of the bill of material, while part 4 and 5 are at the second and third levels respectively.

3. *Additional renewable resource requirements to perform an operation (tools, fixtures, personnel, etc.).* Processing an operation may require several resource types. Several units of each resource type can also be required. However, only one machine can be used to process an operation. For instance executing an operation O may require machine M and three other resource types: one unit of resource type X, one unit of resource type Y and 2 units of resource type Z, where X can be an operator with certain special skills, and Y and Z two different tools. The proposed system assumes that there is considerably more competition for machines than for other resources. So machine resources are treated differently as it will be shown later. This assumption is real for most production plants.

4. *Alternative processing plans for each part.* Sometimes a part can be processed by two or more technologically different sequence of operations, that is, by two or more different sequence of machine types.

In order to facilitate the modeling of alternative processing plans the part process is divided into a sequence of subprocesses. There is no feasible sequence of operations that can process the part without the use of at least one operation of each subprocess. So a unique sequence of subprocesses determines all possible processing plans for a part. Each subprocess can be performed by alternative operation routes. Each route is formed by a sequence of machine cells, each machine cell corresponding to an operation. A machine cell is a set of identical machines. However, if a set of identical machines processes some operations with sequence dependent setups, then each one of these machines is modeled as a machine cell. This modeling trick is adopted in order to simplify the active schedule generation algorithm developed in section 4.2. The user must provide the part processing plans to the system. When dividing a process into subprocesses, the user must maximize the number of subprocesses, once it minimizes the total number of alternative subprocess routes, and therefore enhances the performance of the schedule generation algorithm (see section 4.2). Note that the number of alternative subprocess routes is defined as the number of subprocesses routes minus one. The following example illustrates these concepts:

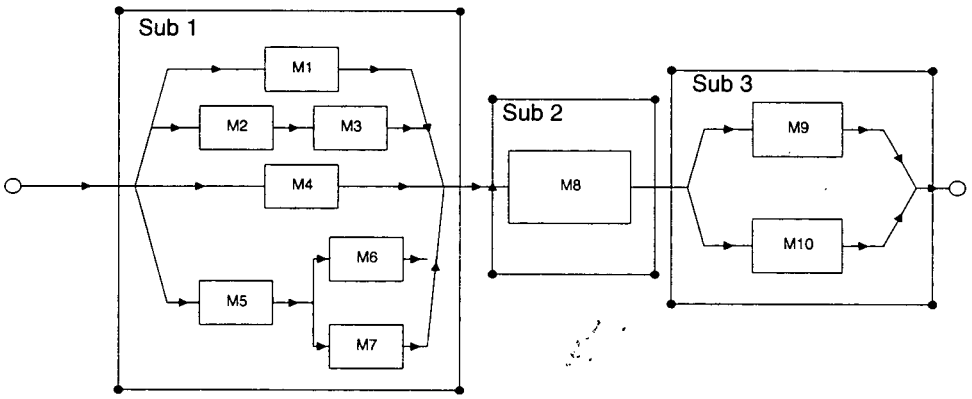


Figure 4.2. Part Process Plan with Alternative Subprocess Routes

In the above diagram, the part process is divided into the three subprocesses Sub 1, Sub 2, Sub 3. The number of routes and alternative routes in Sub 1 are five and four respectively, in Sub 2 are one and zero, and in Sub 3 are two and one. For instance, a possible processing plan can be executed by the following sequence of machine cells: M5, M7, M8, M9. In this processing plan the route M5, M7 was chosen

among the subprocess 1 routes, the subprocess 2 has only one possible route (M8) and M9 was chosen as the subprocess 3 route.

Note that it is assumed that different subprocess routes are defined only by different sequences of machine cells. The other resources (e.g., staff, tools) required to process the operation are not allowed to vary either in type or in quantity. That is, resources other than machines do not generate alternative subprocess routes. Nevertheless, the proposed framework will also work if this assumption is relaxed by defining each route as a sequence of resource sets (instead of machine cells), each resource set corresponding to an operation.

5. *Alternative resources for each operation.* Since there can be more than one unit of each resource type in the production plant, there can exist alternative sets of resources to process an operation.

6. *Machine, tool, fixture and staff calendars (e.g., preventive maintenance, staff training, etc.).* Machines have scheduled breaks for maintenance and cleaning, tools can need sharpening, personnel requires vacation and training. The resources in the production environments are not operational 100% of the time. So, the calendar of each resource must be taken into account.

7. *Ready times for raw material and externally produced parts, represented by job and operation ready times.* Very often processing an operation requires raw materials or components not manufactured in the plant, but purchased from suppliers. These materials and components may be available only after a specific point in time. The inventory control system must provide the availability dates of these items to the scheduling system, which in turn interprets them as operation and job ready times, once a job or operation can not be initiated unless all raw materials and components required are available.

8. *Sequence dependent setup times.* An operation presents sequence dependent setup time when its setup time depends on which operation was previously processed on the machine. Therefore a setup table must exist for each of these operations.

9. *Batch overlap, i.e., successive operations of the same batch of parts being performed simultaneously.* For instance, let the batch size of a part be equal to 40 and let the manufacturing of this part requires 2 operations. Under certain conditions (see section 4.2) the second operation can be initiated after only a fraction of the 40 parts has completed the first operation. For instance, each time 10 parts finish the first operation they are transported to another machine cell to the second operation. Although batch overlap is allowed, concurrent processing of the same operation on more than one machine is not allowed, i.e., batch splitting is not allowed. It means that an operation can start without all parts of the batch has completed the preceding operation, but only one resource set (with only one machine) is chosen to process an operation. Also, batch preemption is not allowed, i.e., once an operation execution starts, it must not stop until the complete batch has been processed. The overlap of the last operation of a job can be used to model partial shipping. Two different overlap policies can be applied, depending on the requirement of parts, to initiate the setup. These issues will be addressed later again in section 4.2.

10. *Any regular multiobjective evaluation function, i.e., any combination of regular performance measures.* The multiobjective function considers combinations of regular performance measures only, because the moves in the local improvement algorithm are based on active chain manipulation and the recombination operator in the GA is embedded in an active schedule generator. Also, it is assumed that the multiobjective function is provided by the user. A regular performance measure can not be improved by delaying the completion time of any job. The most commonly used regular performance measures can be grouped in the following basic sets, each set formed by equivalent measures (as shown by French, 1982):
    e) Mean flow time, Mean completion time, Mean waiting time, Mean lateness
    f) Maximum completion time (makespan), Mean machine idle times, Total machine idle time, Mean number of jobs being processed per unit time
    g) Maximum tardiness
    h) Maximum lateness
    i) Maximum flow time
    j) Mean tardiness

One can easily prove that any linear combination of these criteria are also regular. Only linear combinations will be considered here, although any other combinations which result in a regular measure can also be used. Mean and maximum eariieness are examples of important non regular performance measures. The inclusion of these criteria in the objective function is not prohibited. However, as mentioned above, the system is not well prepared to minimize non-regular measures.

As it will become evident in next sections, the extra computational difficulties introduced by the above characteristics are not simply additive. In many cases they interact with one another and a considerably large amount of computational time must be spent to consider all of these constraints simultaneously.

The proposed system is composed of three basic modules: the modified schedule generator algorithm, the local hill climbing procedure and the hybrid genetic algorithm. The system works as follows: a set of initial solutions is obtained by modified active and non-delay schedule generation algorithms. Each initial solution is enhanced by a local improvement algorithm. Then a genetic algorithm hybridized with a local hill climbing procedure is applied to the set of local optimum schedules. The genetic algorithm is dependent on the schedule generation algorithm, since its crossover operator is embedded in the decoding procedure performed by the schedule generator. An overview of the proposed scheduling system is shown in the diagram below

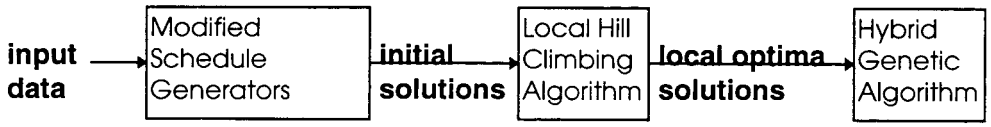


Figure 4.3. Scheduling System Diagram

The next three sections describes the three basic system modules.

4.2. THE MODIFIED ACTIVE AND NON-DELAY SCHEDULE GENERATION ALGORITHMS

As described in chapter 3, an active schedule is a schedule where no operation can start earlier without delaying any other operation. In a non-delay schedule no

machine is kept idle if it could begin processing an operation (Baker, 1974). The non-delay schedule set is a subset of the active schedule set which is a subset of the semi-active schedule set. It was shown that all optimal schedules related to any regular measure are active, but not necessarily non-delay (French, 1982). The active schedule generation algorithm developed by Giffler and Thompson (1960) is able to generate all possible active schedules for classical job shop scheduling problems. A large amount of research has been reported on the use of deterministic or probabilistic dispatching rules in active and non-delay schedule generation algorithms proposed by Thompson and Giffler (1960). These schedule generation algorithms have also been used to generate good initial solutions in applications with genetic algorithms or other local search procedures for classical JSSP (Mattfeld, 1996).

In this section a modified active schedule generation algorithm which deals with the real job shop scheduling problems described in section 4.1 is proposed. Afterwards, some simplifications in the basic algorithm are presented to enhance its computational performance and a modified non-delay schedule generation algorithm is developed. Finally, a set of heuristics to be embedded in these modified schedule generation algorithms are proposed.

4.2.1. Basic Relations, Calculations and Operations

We present here some basic relations, operations and calculations commonly used in the algorithms developed in this work.

Availability of resources to process an operation

Let Q_R be the number of resources of type R in the system. Let RS_u be the set of resources required to process operation u - RS_u = {(R, q_R), where q_R units of resource type R are required to process operation u}. The availability of resources to start processing operation u at time t is verified by the availability of at least q_R out of Q_R resources of type R from t to $t+t_u$, $\forall$ (R, q_R)$\in$ RS_u, where t_u is the total batch processing time of operation u. Note that the resource calendars (including maintenance, vacation, etc.) must be taken into account.

Precede (prec) and succeed (suc) operators

The *precede* (prec) and *succeed* (suc) operators determine respectively all the operations that directly precede and succeed an operation of a job, considering prespecified subprocess routes (i.e., subprocess routes already chosen) and several subassembly levels.

If u is the first operation of part p, with p not in the first level of the bill of material of job j, then prec(u) results in a set of more than one operation, each of these operations being the last operation of a part preceding part p in the bill of material of job j. If u is the first operation of part p, with p in the first level of a bill of material (i.e., p has no preceding parts), then prec(u)=∅. If u is not the first operation of part p, prec(u) results in the operation that technologically precedes u in the processing plan of p.

If u is the last operation of job j, then suc(u) = ∅. If u is not the last operation of job j, than suc(u) results in the operation that technologically succeeds u in the processing plan of job j.

In a very similar way the *succeed* and *precede* operators are defined in the subprocess and part domains.

Determination of batch overlap time (to_u)

The batch overlap time of operation u is defined as the minimum amount of time (after operation u has begun) required to start processing the operation succeeding u on job j (suc(u)) and will not interrupt it due to unavailability of parts. Let α_u be the minimum transport batch of operation u, which is defined as the minimum number of parts from a batch of size n that can be transported after the completion of operation u. The concept of minimum transport batch usually corresponds to the concept of unit load in material handling theory.

In order to simplify the calculation of to_u, let the minimum transport batch α_u be such that the ratio n/α_u is an integer. This theoretical assumption is adopted here only for clarity purposes. Let ts_u be the setup time of operation u and top_u the unit execution time of u. The following two batch overlap policies are defined:

Policy 1. An operation setup is initiated only after at least one minimum transport batch of the preceding operation has arrived at the machine input buffer. It means that an operation setup is initiated only after parts to be processed become available. This policy must be adopted when the setup time is not very large or the time data variance is high There are two cases, as shown below:

a) $to_u = ts_u + \alpha_u * top_u$, if $t_{suc(u)} > \alpha_u * top_{suc(u)} + (n - \alpha_u) * top_u$

b) $to_u = t_u - t_{suc(u)} + \alpha_u * top_{suc(u)}$, if $t_{suc(u)} \leq \alpha_u * top_{suc(u)} + (n - \alpha_u) * top_u$

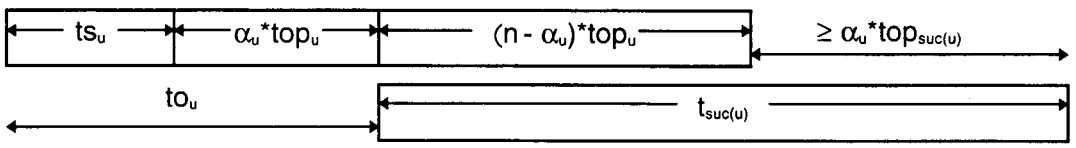
Policy 2. An operation setup can be initiated without any part be available, given that no time gap occurs between the setup and the beginning of operation execution. Again, there are two cases:

a) $to_u = ts_u + \alpha_u * top_u - ts_{suc(u)}$, if $top_{suc(u)} > top_u$

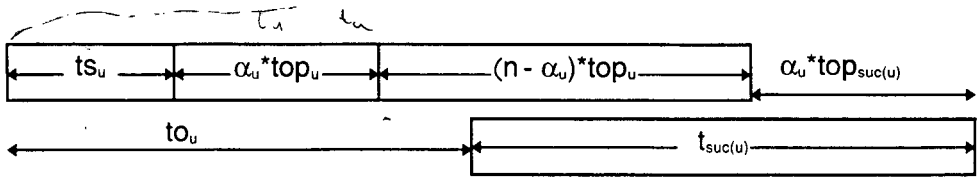
b) $to_u = t_u - t_{suc(u)} + \alpha_u * top_{suc(u)}$, if $top_{suc(u)} \leq top_u$

These four situations are described in the following diagrams:

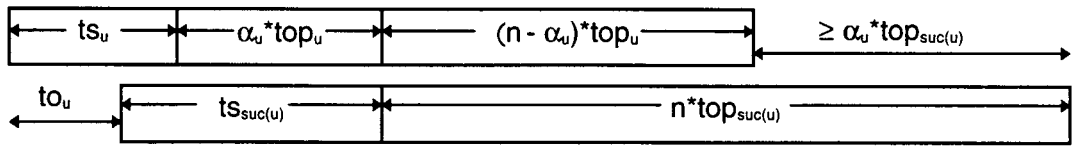
Policy 1 / case a:



Policy 1 / case b:



Policy 2 / case a:



Policy 2 / case b:

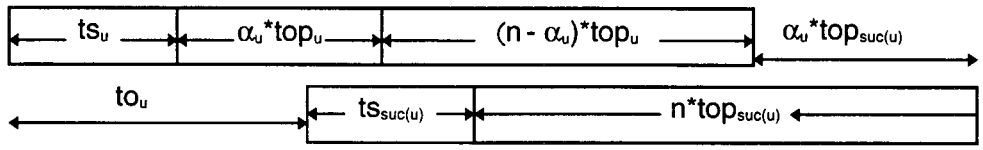


Figure 4.4. Overlap Policies

Note that to_u under policy 2 can be negative, meaning that a time demanding setup can start before the setup of the preceding operation has started. In order to avoid this situation we consider $to_u = \max \{to_u, 0\}$.

When u is the last operation of a subprocess s, then the value of to_u depends on the route chosen for the following subprocess (suc(s)). Moreover, if u or suc(u) are sequence dependent setup operations, then to_u will also be sequence dependent.

Partial shipping can easily be modeled by allowing overlap in the last operation of a job. Here, as in the rest of this work, the transportation times were not considered. As long as transportation devices are always available, only as few modifications in the to_u equations are required to include the inter-station transportation times.

Determination of operation earliest start time (l_u)

The earliest start time of an operation u (l_u) is obtained by determining the minimum l_u, such that:

- $l_u \geq rj$, rj is the job ready time
- $l_u \geq ro_u$, ro_u is the operation ready time
- $l_u \geq \max_{prec(u)} \{l_{prec(u)} + to_{prec(u)}\}$, to_u is the batch overlap time

- There is availability of resources to process the operation from l_u to (l_u+t_u), where t_u is the total batch processing time of operation u

After the introduction of these concepts, we can describe the modified active and non-delay schedule generation algorithms.

4.2.2. Description of the Schedule Generation Algorithms

The modified active scheduling generation algorithm (MASGA) will now be presented. The algorithm can be viewed as a generalization of the classical active schedule generation algorithm proposed by Giffler and Thompson (1960), and it is a robust framework for heuristic-based scheduling research in real production environments as the one described in section 4.1.

An active schedule is now defined as a schedule where no operation can start earlier, *even using another set of resources*, without delaying other operation.

The following symbols are used in the algorithm:

PS_i = partial schedule at stage i, corresponding to the set of operations already scheduled at stage i.
S_i = set of schedulable operations at stage i, corresponding to the set of operations at stage i for which all the preceding operations are in PS_i.
RCO_{ϕ^*} = set of resources which can be used to complete operation u* at time ϕ^*.
NP_{jp}= Number of parts directly preceding part p in the bill of material of job j
ITR_u = Set of resources selected to process operation u

The MASGA works as follows:

1) $i = 0$; $PS_i = \varnothing$; $S_i = \varnothing$.
2) For each part p such that $NP_{jp}=0$, randomly select exactly one route for the first subprocess of p and insert the first operation of the chosen route in S_i .

3) For all operations $u \in S_i$ such that u is the last operation of a subprocess and suc(u) $\neq \varnothing$, randomly select one route among the alternative routes of the succeeding subprocess (if it has not been already selected).
4) Determine the minimum operation completion time when overlap is allowed $\phi^* = \min_{u \in Si} \{l_u + to_u\}$, where l_u and to_u are obtained as previously shown considering as "prespecified routes" the ones selected in steps 2 and 3.
5) Let u^* be the operation related to ϕ^*. Determine RCO_{ϕ^*}, the set of all resources that can be used to complete operation u^* at time ϕ^*. That is, for all types of resources required to process u^*, include in RCO_{ϕ^*} the resource units available from l_{u^*} thru $\phi^* = l_{u^*} + t_{u^*}$, without violating resource calendars.
6) Determine the set $S_i' \subseteq S_i$ such that $S_i' = \{ u / u \in S_i, l_u \leq \phi^*$, and processing u from l_u to $l_u + t_u$ can use at least one resource belonging to $RCO_{\phi^*} \}$.
7) Randomly select an operation $u^{(1)}$ from S_i' to be scheduled next.
8) Randomly select the resource set $ITR_{u(1)}$ to process operation $u^{(1)}$ from time $l_{u(1)}$ to $l_{u(1)} + t_{u(1)}$.
9) Determine $tdo_{u(1)}$, where $tdo_{u(1)}$ is the decrease in $l_{u(1)}$ caused by not considering resource requirements other than machine to process operation $u^{(1)}$. This variable will be used by the local hill climbing algorithm.
10) Form PS_{i+1} by adding $u^{(1)}$ to PS_i. Form S_{i+1} by removing $u^{(1)}$ from S_i.
11) If the last scheduled operation $u^{(1)}$ is the last operation of a part p and suc(p) $\neq \varnothing$, then

    $Np_{jsuc(p)} = Np_{jsuc(p)} - 1$.

    If $Np_{jsuc(p)} = 0$ then add $suc(u^{(1)})$ to S_{i+1},

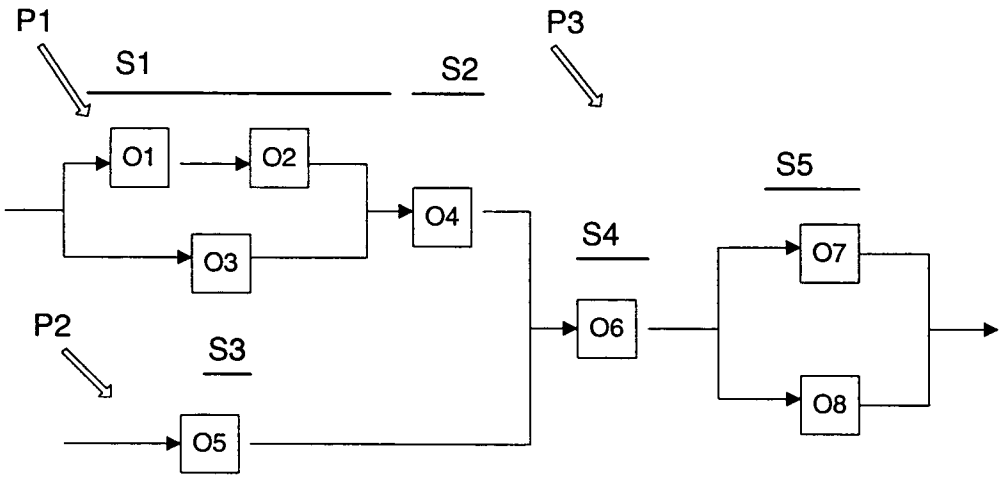
    else add $suc(u^{(1)})$ to S_{i+1}
12) $i = i + 1$.
13) If $S_i \neq \varnothing$ return to step 3; else stop and calculate the evaluation function value.

Note that the determination of $tdo_{u(1)}$ is done only to facilitate the starting of the local improvement procedure (see section 4.3). The following simple example illustrates the reasoning of the MASGA:

Example 4.1.

Two jobs (products) must be scheduled in a job shop. The first job is composed by three parts and the other job has only one part. The flowcharts below show the routing structure of jobs (Ji), parts (Pi), subprocess (Si), routes (Ri) and operations (Oi) provided by the process planning:

J1:



J2:

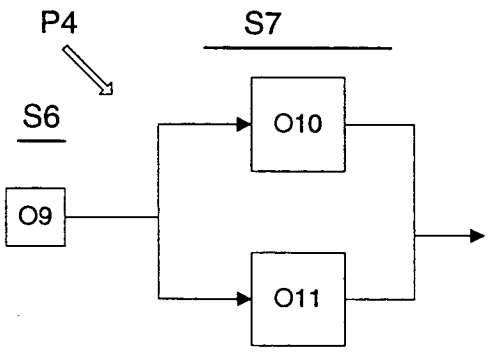


Figure 4.5. Job Structures and Process Plans of Example 4.1

Job J1 is composed of three parts (P1, P2, and P3), part P3 being the succeeding part of parts P1 and P2. One unit of part P1 and one unit of part P2 are required to produce one unit of part P3 (Job J1). Therefore, in this example, the batch sizes of parts P1, P2 , P3, and job J1 are identical. Part P1 is composed of two subprocesses (S1 and S2). There are 2 subprocess routes for subprocess S1, one requiring operations O1 and

O2, and the other requiring only operation O3. The rest of the flowcharts are interpreted similarly. Supposing that all job and operation ready times are zero and that there is no sequence dependent setup operation, the following table provides some important problem data:

Job (batch size)	Part	Subprocess	Route	Operation	Operation setup time	Unit operation execution time	Minimum transport batch
J1 (10)	P1	S1	R1	O1	5	1.5	10
				O2	7	2.0	5
			R2	O3	3	3.0	5
		S2	R3	O4	5	1.0	5
	P2	S3	R4	O5	8	1.5	2
	P3	S4	R5	O6	7	2.5	2
		S5	R6	O7	4	2.0	2
			R7	O8	2	0.5	5
J2 (15)	P4	S6	R8	O9	9	4.0	3
		S7	R9	O10	2	1.5	3
			R10	O11	6	2.0	5

Table 4.1. Operation Related Data of Example 4.1

The production environment is composed of 3 machine cells, the first containing 2 machines and the others containing one machine each. Five other resource types are also used. The description of all resources used in this problem is shown in table 4.2, where Qij is the jth unit of resource type i. The start time and duration of scheduled breaks (for maintenance, training, etc.) for each resource in the scheduling horizon are also displayed.

Resource	Description	Scheduled break (start time, duration)
Q11	Machine of type 1	-
Q12	Machine of type 1	(30, 10)
Q21	Machine of type 2	(15, 15)
Q31	Machine of type 3	-
Q41	Operator of type 4	(0, 10), (80,20)
Q42	Operator of type 4	(50, 15)
Q51	Operator of type 5	-
Q52	Operator of type 5	(150,20)
Q53	Operator of type 5	(45,20)
Q61	Tool of type 6	-
Q62	Tool of type 6	(50,30)
Q71	Tool of type 7	(12,10)
Q72	Tool of type 7	-
Q73	Tool of type 7	(65,25)
Q81	Tool of type 8	(160,10)

Table 4.2. Resource Related Data of Example 4.1

Next table describes the resource requirement of each operation:

Operation	Machine type	Other resources (type, quantity)
O1	1	(4,1), (8,1)
O2	3	(5,1), (7,1)
O3	2	(5,1), (7,2)
O4	1	(4,1)
O5	1	(4,1),(6,1)
O6	3	(8,1)
O7	2	(5,1)
O8	1	(4,1)
O9	2	(5,1),(8,1)
O10	3	(5,1),(6,1)
O11	1	(4,1)

Table 4.3. Resource Requirement Data of Example 4.1

Using overlap policy 1, the algorithm works as follows:

Step	PS_i SCHEDULE PARZIALE	S_i Oj DI OPERAZIONI GIÀ SCHEDULE	Subprocess routes selected	u^*/ϕ^* OP/TERMINO	RCO_{ϕ^*} Oj DI REC. CHE PUÒ SOV UTILIZZO	S_i'	$u^{(1)}$ / $ITR_{u(1)}$ / tdo
1	∅	∅	-	-	-	-	-
2	∅	{O3,O5,O9}	R2,R4,R8	-	-	-	-
3	∅	{O3,O5,O9}	R3,R5,R9	-	-	-	-
4-5	∅	{O3,O5,O9}	-	O5/11	{Q11,Q12,Q42, Q61,Q62}	-	-
6	∅	{O3,O5,O9}	-	O5/11	{Q11,Q12,Q42, Q61,Q62}	{O5}	--
7-11	{O5}	{O3,O9}	-	-	-	-	O5/{Q11,Q42,Q62
3	{O5}	{O3,O9}	-	-	-	-	-
4-5	{O5}	{O3,O9}	-	O3/53	{Q21,Q51,Q52, Q71,Q72,Q73}	-	-
6	{O5}	{O3,O9}	-	O3/53	{Q21,Q51,Q52, Q71,Q72,Q73}	{O3,O9}	-
7-11	{O5,O9}	{O3,O10}	-	-	-	-	O9/{Q21,Q52,Q81
3	{O5,O9}	{O3,O10}	-	-	-	-	-
4-5	{O5,O9}	{O3,O10}	-	O3/122	{Q21,Q51,Q52, Q53,Q71,Q72,Q73}	-	-
6	{O5,O9}	{O3,O10}	-	O3/122	{Q21,Q51,Q52, Q53,Q71,Q72,Q73}	{O3,O10}	-
7-11	{O5,O9,O3	{O10,O4}	-	-	-	-	O3/{Q21,Q52,Q71,Q
3-11	{O5,O9,O3, O10}	{O4}	-	O10/123.5	{Q31,Q51,Q53, Q61,Q62}	{O10}	O10/{Q31,Q53,Q6
3-11	{O5,O9,O3, O10,O4}	{O6}	-	O4/142	{Q11,Q12,Q41, Q42}	{O4}	O4/{Q12,Q42}/(
3-11	{O5,O9,O3, O10,O4,O6}	{O8}	R7	O6/196	{Q31,Q81}	{O6}	O6/{Q31,Q81}/2
3-13	{O5,O9,O3, O10,O4,O6,O8}	∅	-	O8/209	{Q11,Q12,Q41, Q42}	{O8}	O8/{Q11,Q41}/(

Table 4.4. Solution Procedure Table of Example 4.1

The total processing time, overlap time and earliest start time of the scheduled operations are shown below, together with the color legend used in the Gantt chart that follows. In this example a very poor solution was obtained

Operation	Total Processing Time	Overlap Time	Start Time	Color
O3	33	23	99	
O4	15	10	132	
O5	23	11	0	
O6	32	26	170	
O8	7	7	202	
O9	69	49	30	
O10	24.5	24.5	99	

Table 4.5. Time Results of Example 4.1

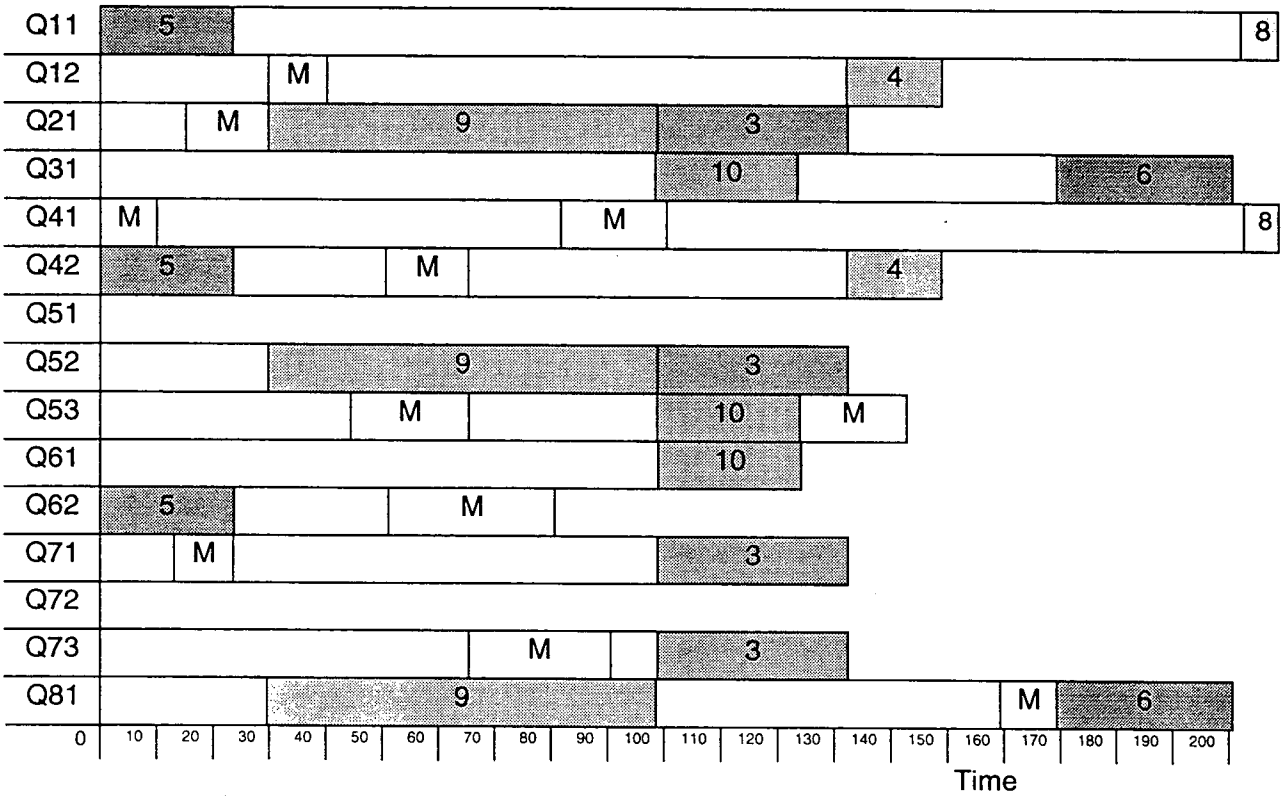


Figure 4.6. Gantt Chart - Example 4.1

If sequence dependent setup operations do not occur, the MASGA describe above is able to generate all active schedules (and hence all optimal solutions) in job

shop scheduling problems considering all real production environment constraints and alternatives described in section 4.1. Moreover, all schedules generated are active.

However, if sequence dependent setup operations are present the overlap time of operation u (to_u) in step 4 can not be precisely determined, and hence ϕ^* can be under or overestimated. This will happen when suc(u) is a sequence dependent setup operation. if so, its setup time ($ts_{suc(u)}$) can not be determined yet, once suc(u) is not even in set S_i. As $ts_{suc(u)}$ is used in the calculation of to_u and to_u used in the calculation of ϕ^*, ϕ^* can not be exactly determined. Three different strategies can be taken to solve this impasse:

1) choose the shortest setup time in the setup table of suc(u) and use this value in the calculation of to_u. This will maximize to_u and ϕ^* as well. As consequence a few semi active schedules (but not active schedules) can be generated. Of course the algorithm will still be able to generate all optimal solutions.

2) choose the largest setup time in the setup table of suc(u) and use this value in the calculation of to_u. This will minimize to_u and ϕ^*. Therefore a few active schedules will never be generated and the capability of generating all optimal schedules will not be hold anymore. Note that all the schedules will be active.

3) choose the average or default value for the setup time of suc(u). This does not guarantee that all optimal solutions will be generated or that all generated schedules will be active. However it reduces the number of non active schedules (compared to strategy 1), and also decreases the probability of not being able to generate the optimal solution (compared to strategy 2). As long as the MASGA described will be used here to generate a set of initial solutions, this third strategy will be adopted.

The setup times of suc(u) chosen above are only for the calculation of ϕ^*. The exact setup time value will only be known when suc(u) is scheduled.

Steps 5 and 6 of the MASGA can be modified to include only machines in RCO_{ϕ^*}. This would disable the algorithm from generating all feasible active schedules, but would speed up the procedure. The modifications in steps 5 and 6 are the following:

5) Let u* be the operation related to ϕ^*. Determine RCO_{ϕ^*}, the set of all *machines* that can be used to complete operation u* at time ϕ^*. That is, if c is the machine type required to process u*, then include in RCO_{ϕ^*} all *machines* of type c available from I_{u^*} thru $\phi^*=I_{u^*}+t_{u^*}$, without violating *machine* calendars.
6) Determine the set $S_i' \subseteq S_i$ such that $S_i' = \{u / u \in S_i, I_u \leq \phi^*$, and processing u from I_u to $I_u + t_u$ can use one of the *machines* belonging to $RCO_{\phi^*}\}$.

When only machines are considered in RCO_{ϕ^*}, the MASGA is said to be in its *simplified generation mode*, as opposed to the *complete generation mode* that also includes resources other than machines in RCO_{ϕ^*}. The use of simplified generation mode probably does not affect the solution quality very much, since the competition level for machines is considerably higher than for other resources.

Better results have been obtained by using dispatching rules in non-delay schedule generation algorithms, instead of active schedule generation algorithms (Baker, 1974). Slight modifications in steps 4 and 6 can transform MASGA in a non-delay algorithm. They are:

4) Determine the minimum operation start time $\phi^*=\min_{u \in Si} \{I_u\}$, where I_u is obtained as previously shown.
6) Determine $S_i' = \{ u / u \in S_i, I_u=\phi^*$, and processing u from I_u to $I_u + t_u$ can use at least one resource belonging to $RCO_{\phi^*}\}$.

Both active and non-delay schedulers using complete and simplified generation modes will be used in the simulations described in chapter 5. Following the modified non-delay schedule generation algorithm (MNSGA) adopting the simplified generation mode is presented:

1) $i = 0$; $PS_i = \varnothing$; $S_i = \varnothing$.
2) For each part p such that $NP_{jp}=0$ randomly select exactly one route for the first subprocess of p and insert the first operation of the chosen route in S_i .

3) For all operations $u \in S_i$ such that u is the last operation of a subprocess and suc(u) $\neq \varnothing$, randomly select one route among the alternative routes of the succeeding subprocess (if it has not been already selected).
4) Determine the minimum operation start time $\phi^* = \min_{u \in Si} \{l_u\}$, where l_u is obtained as previously shown.
5) Let u* be the operation related to ϕ^*. Determine RCO_{ϕ^*}, the set of all *machines* that can be used to complete operation u* at time ϕ^*. That is, if c is the machine type required to process u*, then include in RCO_{ϕ^*} all *machines* of type c available from l_{u^*} thru $\phi^* = l_{u^*} + t_{u^*}$, without violating *machine* calendars.
6) Determine the set $S_i' \subseteq S_i$ such that $S_i' = \{ u / u \in S_i, l_u = \phi^*$, and processing u from l_u to $l_u + t_u$ can use at least one machine belonging to $RCO_{\phi^*} \}$.
7) Randomly select an operation $u^{(1)}$ from S_i' to be scheduled next.
8) Randomly select the resource set $ITR_{u(1)}$ to process operation $u^{(1)}$ from time $l_{u(1)}$ to $l_{u(1)} + t_{u(1)}$.
9) Determine $tdo_{u(1)}$, where $tdo_{u(1)}$ is the decrease in $l_{u(1)}$ caused by not considering resource requirements other than machine to process operation $u^{(1)}$. This variable will be used by the local hill climbing algorithm.
10) Form PS_{i+1} by adding $u^{(1)}$ to PS_i. Form S_{i+1} by removing $u^{(1)}$ from S_i.
11) If the last scheduled operation $u^{(1)}$ is the last operation of a part p and suc(p) $\neq \varnothing$, then

    $Np_{jsuc(p)} = Np_{jsuc(p)} - 1$.

    If $Np_{jsuc(p)} = 0$ then add $suc(u^{(1)})$ to S_{i+1},

    else add $suc(u^{(1)})$ to S_{i+1}
12) $i = i + 1$.
13) If $S_i \neq \varnothing$ return to step 3; else stop and calculate the evaluation function value.

In the schedule generation algorithms proposed in this section, selections in steps 2, 3, 7, and 8 were made at random. However, one can also make these selections using deterministic or probabilistic heuristics. Of course, if only deterministic heuristics are used the algorithm will be able to generate only one solution.

4.2.3. Heuristics Embedded in the Modified Schedule Generation Algorithms

The following heuristics were developed to aid the selections in steps 2, 3, 4, and 8 of the modified schedule generation algorithms presented in the last section:

Selection of subprocess routes (Route Selection Heuristic)

The selection of subprocess routes in steps 2 and 3 are based on two different objectives: (1) balance resource utilization, avoiding highlighted bottlenecks, (2) choose efficient routes, i.e., routes associated with low processing times.

Let P(rt,s,i) be the probability of choose route rt to execute subprocess s at stage i, where a stage corresponds to the selection of a route in the MASGA. Let **RU**(R,i) be the expected utilization time of a machine from cell R at stage i; Q_R be the number of machines in cell R; MC_u the machine cell required to process operation u; t_u the operation u processing time; N the total number of operations of all jobs; TR_s the number of alternative routes of subprocess s. Then:

$$RU(R,i)= \Big(\sum_{\substack{u\in rt\ /\ rt\ chosen \\ MC_u=R}} t_u \quad + \quad \sum_{\substack{u\in rt\ /\ rt\ not\ chosen \\ MC_u=R}} t_u * P(rt,s,i) \Big) / Q_R$$

The expected weighted utilization time of machines in a specific route rt at stage i is given by Rt(rt,i), where:

$$Rt(rt,i) \ = \sum_{\substack{u\in rt \\ MC_u=R}} t_u * RU(R,i)$$

The route rt processing time, Rpt(rt), is simply:

$$Rpt(rt) \ = \sum_{u\in rt} t_u$$

Let $0 \le a \le 1$ be the relative importance between balancing resource utilization and adopting efficient routes. P(rt',s,i) is considered inversely proportional to:

$$a*Rpt(rt') / \sum_{rt=1}^{TR_s} Rpt(rt) \quad + \quad (1-a)*Rt(rt',i) \ / \ \sum_{rt=1}^{TR_s} Rt(rt,i)$$

The method to determine the values of P(rt,s,i) is recursive, since P(rt,s,i) is used to calculate RU(R,i) either. So, at each stage the following convergent problem is solved:

P(rt,s,i) is initialized with P(rt,s,i-1), $\forall$ rt

P0(rt,s,i)=0, $\forall$ rt,

while $\max_{rt}$ |P(rt,s,i)-P0(rt,s,i)|>ERROR do

P0(rt,s,i) = P(rt,s,i);

determine P(rt,s,i) as described above;

Note: $P(rt,s,1)=1/TR_s$

Selection of the operation to be scheduled next

The determination of which operation of S_i' to be scheduled next (dispatching procedure) in step 7 is done totally at random or by using the traditional shortest processing time dispatching rule (SPT).

Selection of the resource set to process an operation (Minimum Gap Heuristic)

The choice of the resource set $ITR_{u(1)}$ to process operation $u^{(1)}$ in step 8 is done in order to minimize time gaps in such resources. So a resource is chosen among others in order to minimize the difference between $I_{u(1)}$ and $I_{u'}+t_{u'}$, where u' directly precedes operation $u^{(1)}$ on this resource.

The performance of these heuristic will be determined in the experimental designs carried out in chapter 5. Note that other approaches can be adopted to make these selections. For instance, decision trees built by inductive learning methods [13] can be used instead of the heuristics proposed above.

Once the initial schedules have been generated, a local improvement procedure is applied to these solutions in order to enhance their evaluation function values. The local improvement algorithm used is described in the following section.

4.3. THE LOCAL IMPROVEMENT PROCEDURE

When dealing with local search algorithms, one must define the neighborhood structure and the related moving operators. The moving operators provide moves from one solution to another in the neighborhood.

As described in chapters 2 and 3, local search procedures like Tabu Search, Simulated Annealing, Threshold Acceptance, and simple Local Iterative Search (Local Hill Climbing) have widely been applied to the classical JSSP, considering makespan as criterion. These applications are based on the disjunctive graph representation, and neighborhood moves are obtained by reversing arcs on the critical path or making other changes on precedence relations in the critical path.

In this section the graph representation is expanded to support real environment constraints. Further, a neighborhood structure based on active chains manipulation is developed to support multiobjective functions and real world constraints. Like the modified schedule generation algorithms, the local search framework developed here is robust. It can be used to implement any local search procedure (as Tabu Search and Simulated Annealing), by only adding the respective control strategy.

Here, a simple Local Hill Climbing Search, also known as Local Iterative Search, will be implemented. In this approach, only moves to better solutions are accepted. The search stops when no improving move is available, that is, when a local minimum has been reached.

The local improvement framework developed takes into account all the real world constraints described in section 1, except alternative subprocess routes and alternative machines. It means that all subprocess routes must be selected before the local hill climbing procedure be applied. Also, each operation must be assigned to a machine before the algorithm starts running, and no other machine can be used to process the operation during the iterative search cycle, that is, when a move is performed the machine used to process an operation is not allowed to change. Nevertheless, assignment of resources other than machines will be allowed to change when moves in the neighborhood are implemented.

Some representation schemes, the basic neighborhood structure, and the local hill climbing algorithm itself are introduced next.

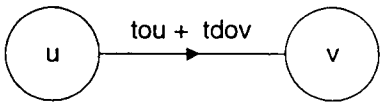
4.3.1. Representation of Real World Constraints in Disjunctive Graphs

A job shop problem is represented by a graph G=(V, A ∪ H). The set of nodes V corresponds to the set of operations, the arc set A connects consecutive operations of the same job, and the set of edges H connects operations processed on the same machine. When the edge set H is transformed into a conjunctive arc set S, a solution is obtained. For practical purpose only the arcs belonging to the Hamiltonian path L_i of each machine i are represented. Arc (v, u)$\in L_i \Leftrightarrow$ operation u is the next operation after v to be processed on machine i, i.e., v<u. The final digraph obtained D = (V, A ∪ L), where L=∪ L_i, represents a particular schedule. If arc (v, u)$\in$ A then u=suc(v). If arc (v,u)$\in$ L then v<u.

Next, this classical schedule representation scheme is extended to include real world constraints. A graph representing a solution of an example problem is also provided at the end of this section. The following arcs are used to represent real job shop scheduling problems:

a) Sequence of operation within a job are represented by arcs (u,v) $\in$ A, where u and v are consecutive operations of the same job. The weight of (u,v) is the sum of to_u (overlap time of operation u) and tdo_v (decrease in the start time of operation v caused by not considering resource requirements other than machine to process v). The weight to_u + tdo_v of arc (u,v) means that operation v can be initiated to_u + tdo_v time units after operation u has started. If v is the first operation of a part p, then there will be as many of these arcs arriving on v as the number of directly preceding parts of part p. These arcs are illustrated below:

Successive operations within a part:



Successive operations in a subassembly:

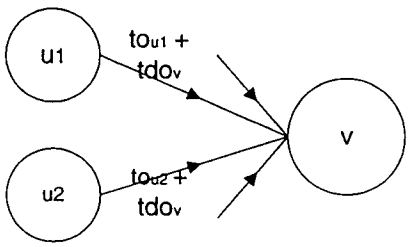


Figure 4.7. Graph Representation of Successive Operations within a Job

b) Job ready times are modeled as the first operation of a job j. Therefore, we create arcs (u,v) $\in$ A with weights corresponding to the sum of job ready time and tdo_v, where u is a dummy node and v is the first operation of part p of job j, given that part p has no preceding part. The arc connecting the source node and u has weight zero. This modeling trick is shown in the following figure, where rj is the ready time of job j, and v1 and v2 are the first operation of parts p1 and p2. Parts p1 and p2 are in the first level of the bill of material of job j, i.e., they are not preceded by any part.

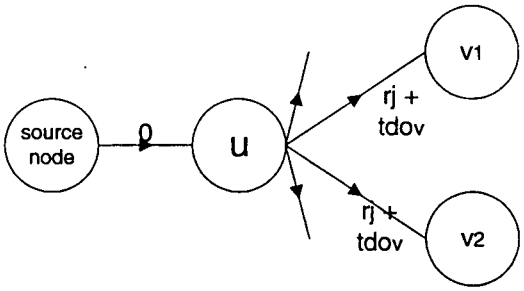


Figure 4.8. Graph Representation of Job Ready Time

c) Similarly, an operation ready time is represented by creating an arc (u,v) $\in$ A of weight corresponding to the sum of ready time of operation v (ro_v) and tdo_v, where u is a dummy node connected to the source node by an arc of weight zero. The following figure illustrates the graph representation of an operation ready:

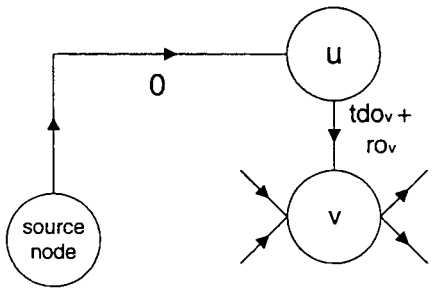


Figure 4.9. Graph Representation of Operation Ready Time

d) Machine capacity constraints are represented by arcs (u,v) $\in$ L, where operation u directly precedes operation v on a machine. The arc weight corresponds to the sum of t_u (total processing time of operation u) and tdo_v, as shown in the diagram below:

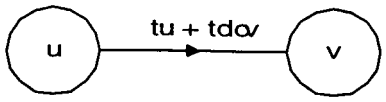


Figure 4.10. Graph Representation of Machine Capacity Constraint

e) Machine calendar constraints are considered by modeling each scheduled maintenance break as an one operation job. The maintenance beginning time is the job ready time, the maintenance duration is the total operation processing time, and the desired maintenance finishing time is the job due date. In the objective function a high weight must be assigned to the violation of this due date in order to enforce the fulfillment of the machine calendar. So we create an arc (u,v) $\in$ A of weight corresponding to the maintenance starting time (rm), where u is a dummy node connected to the source node by an arc of weight zero and v (maintenance operation) is connected to the sink node by an arc of weight equal to the maintenance duration time (tm). The following figure illustrates the graph representation of a machine maintenance operation:

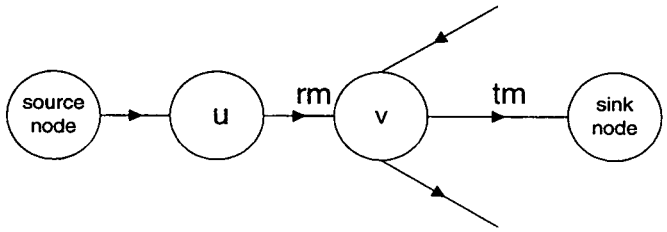


Figure 4.11. Graph Representation of Machine Maintenance

The dummy operations related to machine calendars and ready times must be included in the graph structure, and hence in the topological sorted set associated to the graph. Note that the first sorted set is the final PS_i generated by the schedule generation algorithm, which is represented here by PS_{final}. Each machine m maintenance operation must be inserted in PS_{final} at a position right before the first operation to be processed on machine m after the maintenance operation. The ready times can be placed in the beginning of PS_{final}.

EXAMPLE 4.2. Consider the same data of example 4.1, but also include ready times for job 1 and operation 8. Let A, B, C, D, E, F be respectively the maintenance operation ready time of machine Q12, the maintenance operation of machine Q12, the maintenance operation ready time of machine Q21, the maintenance operation of machine Q21, the ready time of job 1, and the ready time of operation 8. The digraph of figure 4.12 corresponds to a particular solution of the scheduling problem. The dashed arrows correspond to arcs in L and the others correspond to arcs in A.

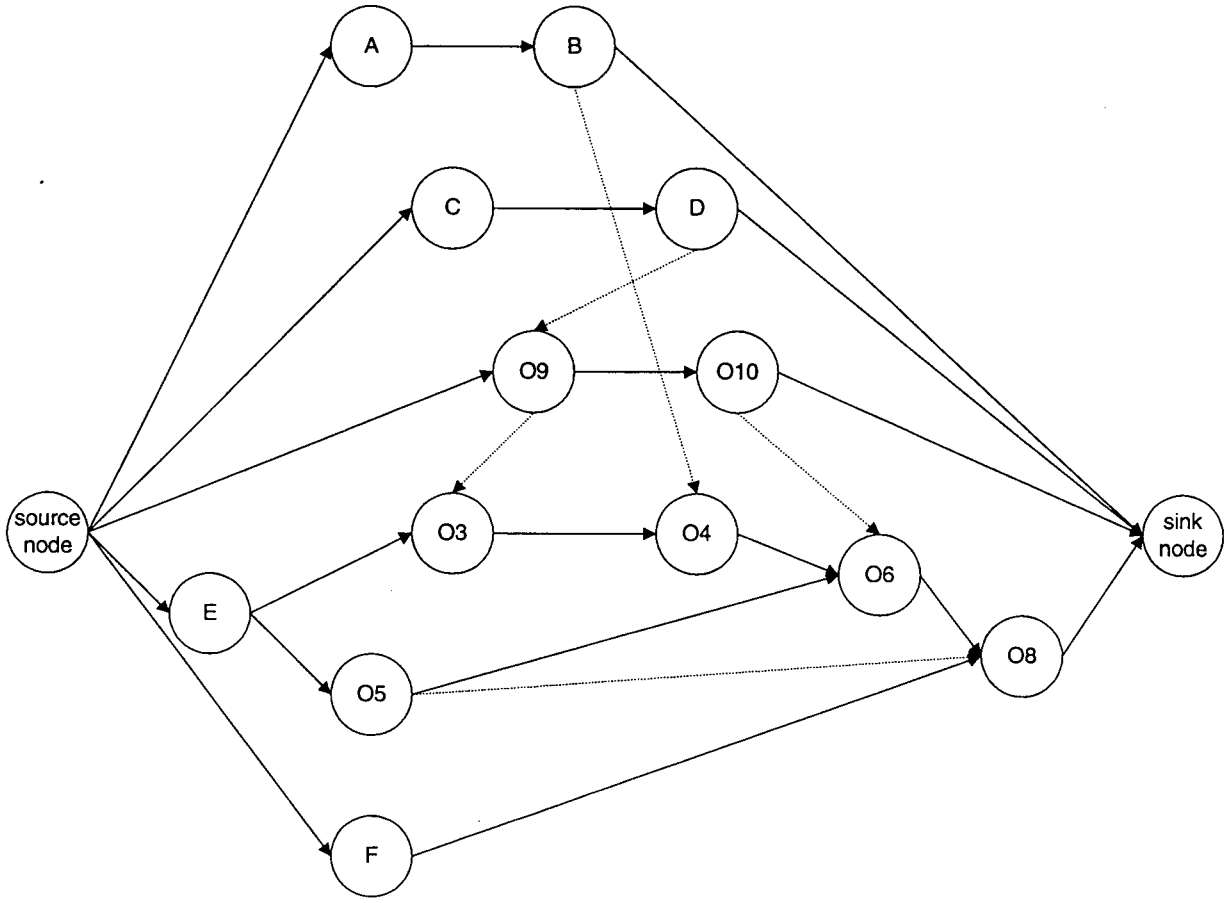


Figure 4.12. Solution Representation by a Digraph - Example 4.2

Once a problem representation scheme was developed a neighborhood structure that allows moving from one solution to another must be defined.

4.3.2. A Neighborhood Structure

Several neighborhood structures, characterized by sets of moves, have been proposed for the classical JSSP with makespan as criterion. Recently, procedures that perform fast makespan estimation for each move and easily recalculate the makespan after a move has been chosen were also developed. These procedures and comparisons among different neighborhoods are reported in (Dell'Amico and Trubian, 1993) and (Mattfeld, 1996).

In this section, a neighborhood is proposed to deal with multiobjective evaluation functions and the already described real world constraints. Moreover, the evaluation function recalculation procedures mentioned in the above paragraph is extended to support such complex environments.

4.3.2.1. Defining a Neighborhood Structure for Real Production Environments

As defined in (Sun et al, 1995) for classical JSSP, an active chain of operation u in a schedule is a chain of operations starting in u and finishing in an operation with earliest starting time equal to zero, and with no time interval between the starting time of an operation and the completion time of the preceding operation in the chain. Two consecutive operations u and v in the chain are such that $(u,v) \in A$ or $(u,v) \in L$. If a schedule is a semi-active one, there is at least one active chain for each operation. It is simple to see that an active chain of operation u corresponds to the head of the operation plus the operation itself. The makespan corresponds to length of the longest active chain, i.e., to the length of the critical path.

As mentioned in section 4.1, the approach proposed here admits any combination of regular measures as objective function. It was proved by Laarhoven et al (1992) that the makespan can only be improved by changing precedence relations among operations on the critical path. Similarly the value of any regular measure can only be improved by changing precedence relations in active chains of last operations of the jobs. Therefore, for each regular measure considered in the multiobjective function is associated a subset of active chains. For instance, the makespan is associated to the longest active chain, the mean tardiness is associated to all active chains of last operations of tardy jobs, and

so on. These active chains are called here *relevant active chains*. After the solution graph has been augmented with the set of dummy operations defined in section 4.3.1, the relevant active chains are extracted from it.

The procedure to build an active chain of an operation is described below, where the variables l_u, to_u, t_u, tdo_u and l_u have been defined in section 4.2:

Starting with the operation and moving backwards, select to include next in the chain one of the following operations (supposing that the most recently included operation is v):

1) u, such that v = suc(u), and $l_u + to_u + tdo_v = l_v$
2) u, such that u<v, and $l_u + t_u + tdo_v = l_v$,

Note that operations u and v above also include the dummy operations used to model machine maintenance and ready times.

If more than one operation can be selected to enter the chain, then decreasing selection priorities are assign to the following operations: (1) job and operation ready times, (2) operations from the same job, (3) operations processed on the same machine. Such selection priorities tend to minimize the neighborhood size and speed up the algorithm.

A block in an active chain (B_i) is a string of consecutive operations that are processed on the same machine i. A neighborhood structure is defined below:

Given a solution D=(V, A ∪ L), its neighborhood N(D) consists of all schedules derived from D by reversing one arc (u,v), where both operations u and v belong to a block B_i of a relevant active chain; and v does not indirectly succeed u in the same job; and either u is the first operation of B_i or v is the last operation of B_i.

The requirement of v not being a indirect successor of u in the neighborhood structure definition above is due to what was named here "flow effect". The flow effect can occur if batch overlap is allowed. Then it is possible for an arc (u,v) in a relevant

active chain be such that u<v and v indirectly succeed u in the same job. For example, let u, w and v be three consecutive operations of the same job (i.e., v=suc(w) and w=suc(u)), and let u and v require the same machine to be processed. The following diagram illustrates a flow effect.

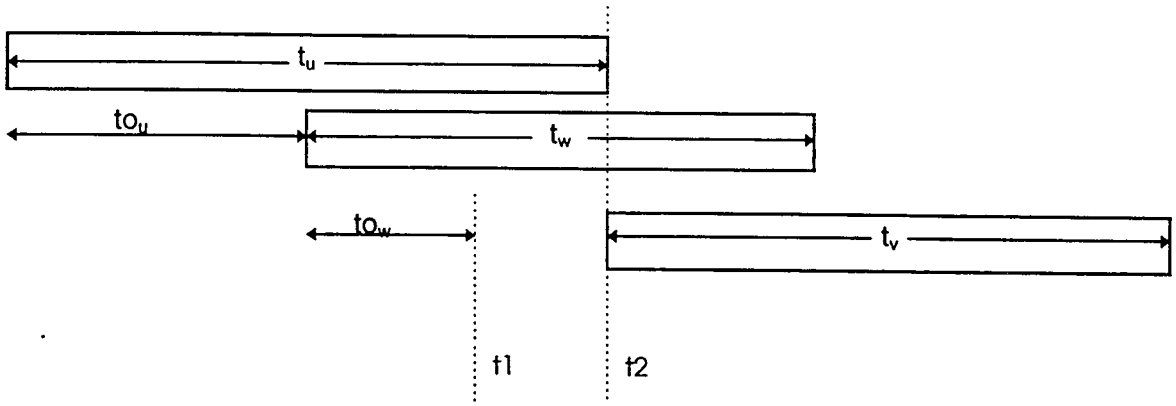


Figure 4.13. Flow Effect Example

If u and v did not require the same machine, v could be initiated at time t1. As t1≤t2 arc (u,v), and not (w,v), is inserted in the active chain. However, (u,v) can not be reversed in a neighborhood move due to technological constraints. Note that there are no two successive operations in a job that are processed on the same machine, because the "two operations" would be modeled as a unique operation.

The following two lemmas are presented and proved:

Lemma 4.1. In the neighborhood N all the available moves lead to feasible schedules (no cycle).

Proof. Let c and d be successive operations on a block of a relevant active chain. The reversal of arc (c,d) can lead to a cycle only if there is a path *p* from suc(c) to prec(d), as can be inferred from figure 4.14. Let v be the last operation on path *p* that is from the same job of c. By the way overlap time was defined, the finishing time of operation v (f_v) is greater than the finishing time of operation c (f_c), i.e., $f_v > f_c$. If path *p* exists, then an operation w such that v<w (w succeed v on the machine) will exist either. As v and w are processed on the same machine the earliest start time of w (l_w) must be greater than or

equal to the finishing time of v, i.e., $l_w \geq f_v$. Therefore $l_w \geq f_v > f_c$. As d is the operation after the path *p* we have $l_d - tdo_d \geq l_w \geq f_v > f_c$, where tdo_d was defined in section 4.2. Thus $l_d > f_c + tdo_d$ and hence arc (c,d) can not be on the active chain.

Lemma 4.2. Considering any combination of regular measures as the evaluation function, the reversal of an arc can lead to an evaluation function improvement only if the reversal move is available in neighborhood N, given that sequence dependent setups and multiple resource requirements are not considered.

Proof. The improvement of a regular performance measure value can only be achieved by reducing the length of relevant active chains. Reversing an arc not belonging to a relevant active chain can not reduce the active chain length. Further, suppose that b, c, d and e are successive operations on a block of a relevant active chain. Thus $l_e = l_b + t_b + t_c + t_d$ (given that only machine is used to process an operation, i.e., $tdo_i = 0$ for all operations i). As shown in figure 4.14, the reversal of arc (c,d) will not affect the start time of operation e (l_e) and hence the active chain length. Therefore a regular performance measure can only be improved by reversing an arc (c,d) of a relevant active chain, where either c is the first operation of a block or d is the last operation of the block.

If the presence of multiple resources for each operation (which generates tdo) and sequence dependent setup operations are considered, the statement that only changes in relevant active chains can improve the evaluation function does not hold anymore. If the weight of an arc in an active chain is sequence dependent, it can be modified by reversing arcs out of the active chain. Similarly, changing the allocation of resources other than machines can alter the value of tdo_u , with u in the active chain, and hence alter the active chain length. However, these effects were disregarded and only moves based on rearranging blocks of relevant active chains were considered.

As for the classical JSSP, the connectivity property does not hold if at least c or d is required to be either the first or the last operation of a block.

As mentioned in last chapter, other neighborhood structures have been proposed for classical job shop scheduling problems. For instance, in (Dell'Amico and Trubian, 1993) a neighbor is obtained by moving an operation to the position closest to the first or

the last operation of its block for which feasibility is preserved. Dell'Amico and Trubian (1993) also considered the reversal of two arcs in one move. These neighborhoods usually reach better local optima. They however require greater computational effort due to their larger sizes. Once considering multiobjective function instead of makespan drastically increases the neighborhood size, these structures will not be used here. Nevertheless, the framework is robust enough to support them.

Depending on the control strategy adopted a number of neighboring solutions must be evaluated in order to select a move. The exact calculation of the makespan value (in classical JSSP) for each of these neighboring solutions require large CPU time. In Dell'Amico and Trubian (1993) is described a fast method to calculate lower bounds for the makespan of each neighboring solution in order to overcome this problem. Unfortunately, a similar procedure is not suitable for real production environments. Even if a fast procedure to calculate lower bounds for an active chain length after a move has been done is available (and it is not available), the multiobjective nature of the evaluation function would invalidate the approach. As an operation usually belongs to several relevant active chains, a promising move regarding to an active chain can produce bad results in other chains. Further, the presence of multiple resources, sequence dependent setups and batch overlap would increase the computational time required to calculate the lower bound and decrease its quality. Thus, an exact calculation of the evaluation function value for every neighboring candidate solutions will be necessary. Similarities between neighboring solutions will be used to enhance this calculation.

4.3.2.2. Performing a Neighborhood Move and Recalculating the Evaluation Function

Let B_i be a block of operations in a relevant active chain to be processed on machine i. The following modifications must be performed in the solution graph D to reverse an arc (c,d) from block B_i:

1) Reverse (c,d) obtaining (d,c).
2) If $(b,c) \in L_i$ exists, then remove (b,c) from L_i and construct arc (b,d).
3) If $(d,e) \in L_i$ exists, then remove (d,e) from L_i and construct arc (c,e).
4) If at least one of the operations c, d, or e has sequence dependent setup times, then update the total processing time of operations c, d, and e, and the overlap time of

operations prec(d), d, prec(c), c, prec(e) and e (if the operations exist). That is, recalculate the values of t_d, t_c, t_e, $to_{prec(d)}$, to_d, $to_{prec(c)}$, to_c, $to_{prec(e)}$, and to_e.

The following figure illustrates the reversal of (c, d) described above:

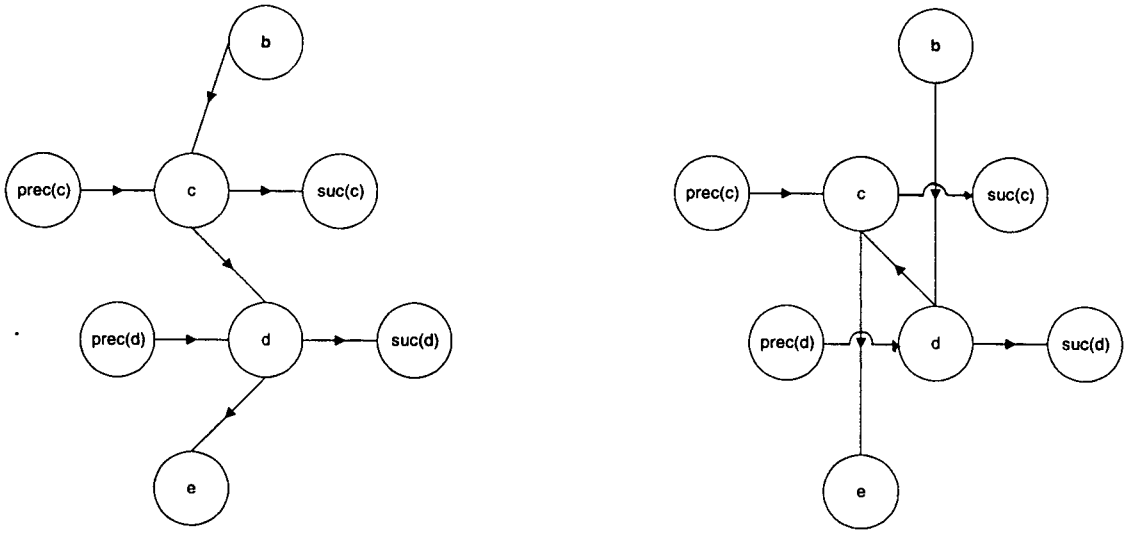


Figure 4.14. Arc Reversal

In the above figure, operations b, c, d and e are processed on the same machine

The graph corresponding to the new solution is obtained by implementing the set of modifications just described in the original graph D. The new solution is represented by D' = (V', A ∪ L'). The arc weoghts in D' are not yet determined because the move could have changed the values of tdo_u.

Before the calculation of the evaluation function value associated to a solution graph F(D), all the operations must be sorted. This procedure can be simplified here by taking advantage of similarities between neighboring solutions.

The first sorted set of operations is provided by a modified schedule generation algorithm and corresponds to the final partial schedule set PS_{final}. As described in Mattfeld (1996) and illustrated in the diagram below, the reversal of (c,d) only affects the sorting of nodes between c and d in PS_{final}. The new sorting is achieved by dividing X into

X' and X", where X' is composed by all operations in X that directly or indirectly precede operation d, and X" = X - X'. This can be achieved by a labeling algorithm (Bradley, Hax and Magnanti, 1977). The sorted operation sets before and after reversing (c,d) are represented by PS_{final} and $PS_{final'}$. The following figure describes the sorted operation sets:

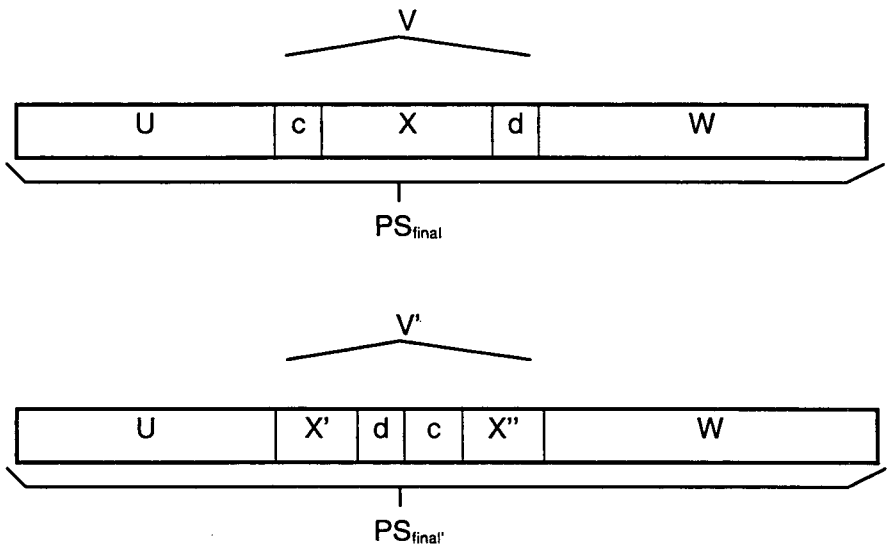


Figure 4.15. Rearranging of Set PS_{final} Due to a Move

After the move has been applied and the operation sorting has been achieved the earliest start time of each operation and the evaluation function value can be determined as follows:

1) The values of l_u (earliest operation start time) and tdo_u (decrease in l_u caused by not considering resource requirements other than machine to process operation u) remain unchanged for all operations u in U. That is, the move does not affect operations in U.

2) From the first operation in V' to the last operation in $PS_{final'}$ do:

   - determine the earliest start time l_u by adding the following two constraint to the set of constraints used to determine l_u in section 4.2.1:

1) $I_u \geq I_v + t_v$, where $(v, u) \in L'$, i.e., $v<u$
2) machine used to process operation u remains unchanged, i.e., assign to process operation u the same machine that was used to process u in the previous solution

- if the same set of resources ITR_u used to process u in the previous solution is available from I_u to $I_u + t_u$ then assign this resource set to operation u; else determine this resource set by the same selection process used in step 8 of the MASGA.
- determine the new value of tdo_u.

3) Once the earliest start time I_u of all operations have been calculated, determine the new evaluation function value.

Remark that the machine used to process an operation u can not change from one solution to another. So the selection of set ITR_u in the above procedure does not include the selection of the machine.

4.3.3. The Local Hill Climbing Framework

Using the graph representation scheme and the neighborhood structure described in last sections any local search procedure can be implemented. This work implements a simple local hill climbing which adopts as control strategy the acceptance of the first improving neighbor. The search stops when a local minimum is reached. A simple hill climbing is used instead of a more sophisticated local search technique (e.g., Tabu Search) because the genetic algorithm will provide the diversification required to lead the search to new regions and escape from poor local minima. Further, the acceptance of the first improving neighbor instead of the best improving neighbor is adopted due to the unavailability of fast multiobjective function estimation methods. The calculation of the exact evaluation function value for all neighboring solutions prior the selection is computational expensive. So, the acceptance of the first improving neighbor is the most efficient strategy, even considering that the number of moves to reach a local optimum using this strategy is about 75% greater than the number of moves required by selecting

the best improving neighbor, as reported in (Mattfeld, 1996) for classical JSSP with makespan as criterion.

The local hill climbing procedure works as follows:

1) Use the schedule generation algorithm to determine an initial feasible solution D.
2) While a local minimum has not been reached do
   2.1) Determine the set of relevant active chains of D
   2.2) Perform a move in the neighborhood of D, i.e., select a solution $D' \in N(D)$
   2.3) Calculate the evaluation function value of the new solution F(D').
   2.4) If $F(D')<F(D)$ then the move is accepted and D=D'.

In step 2.2 of the local hill climbing algorithm an arc $(c,d) \in L$ is selected to be reversed among a set of candidate arcs (see the neighborhood structure definition in section 4.3.2). This selection can be made randomly, sequentially (from the first to the last candidate arc of each relevant active chain), or heuristically.

In this work we propose a simple heuristic to determine the priority of reversing an arc. The heuristic was called "bottleneck heuristic". It takes into account the frequency of occurrence of an arc in the relevant active chains, and the importance of the chains where the arc appears. The bottleneck heuristic works as follows:

1) Determine a weight for each relevant active chain. This weight is directly related to the objective function. If a regular measure has a high weight in the objective function, so it will have its associated active chains. Precisely, the weight of a relevant active chain associated to any maximum-based regular measure (e.g., makespan, maximum tardiness) is the weight of the measure in the objective function. The weight of the relevant active chains associated to mean-based regular measures (e.g., mean flow time, mean tardiness) is the weight of the measure in the objective function divided by the number of jobs that participate in the measure value. For example, consider a 10 job problem, and the following objective function: F = 2*Makespan + 3*Mean flow time. The weight of the longest active chain is 2+3/10=2.3 and the weight of all other nine relevant active chains is 3/10=0.3

2) Select an arc (c,d)$\in$L to be reversed next with probability proportional to the sum of weights of all active chains where (c,d) appears, given that (c,d) is a candidate arc to be reversed according to the neighborhood structure defined in section 4.3.2.

The local hill climbing procedure developed in this section is used not only to improve the initial solutions generated by the modified schedule generation algorithms but it is also hybridized with the genetic algorithm described in next section.

4.4. THE HYBRID GENETIC ALGORITHM

In this section a hybrid genetic algorithm framework to solve JSSP in real production environments is described. The genetic algorithm works in the local optimum domain due to the application of local hill climbing to each new individual generated by reproduction. The decoding procedure, which maps a genotype to a scheduling solution corresponding to a phenotype, and the recombination of individuals are performed simultaneously. The decoding procedure is basically the modified active or non delay schedule generation algorithm, and the crossover operator is embedded in it. The crossover operator is able to combine parent solutions differing not only in the assignment of starting time for operations, but also in the routes assigned to subprocesses and in the resources assigned to operations.

The chromosomes are basically the sorted sets PS_{final}. This representation scheme allows the mapping of all active or non delay schedules, depending on which generation strategy (active or non delay) is adopted. That means that the representation scheme is complete. Although the representation scheme corresponds to the highly redundant "permutation of operations" scheme described in chapter 3, the way the crossover operator is related to the decoding procedure attenuates this redundancy.

The genetic operators and population management strategies used by our hybrid GA are defined next.

4.4.1. Population Management Strategies and Genetic Operators

A classical proportional selection scheme is adopted. The fitness (objective function value) is scaled to the range [0, Fmax - Fmin], where Fmax and Fmin are respectively the maximum and minimum fitness value within the population. This scaling leads to a more severe selection scheme, avoiding a tedious (and hence time expensive) recombination of individuals without significant fitness improvement, that is, avoiding a too slow population convergence.

The number of offsprings is equal to the population size. Also, non-overlapping populations and elitism are adopted. Elitism assures that the best individual is always passed to the next generation. The termination criterion is based on a fixed number of generations. More elaborated termination criteria based on population diversity have not bee proved to be proper for job shop scheduling problems (Mattfeld, 1996). Moreover, a fixed number of generations is more suitable for comparisons in simulation studies.

Two crossover operators are applied: the subprocess route crossover and the basic crossover. The basic crossover operator proposed here is similar to the uniform crossover operator as well as to the active schedule constructive crossover proposed by Park and Park (1995) for classical job shop scheduling problems.

The subprocess route crossover and the basic crossover operations embedded in the decoding procedure are described below:

a) Randomly select k subprocess routes from a parent and S-k subprocess routes from the other, where S is the total number of subprocesses considering all the jobs to be scheduled. Exchange these subprocess routes between the parents to obtain the offspring subprocess routes. This procedure corresponds to the subprocess route crossover. For each new set of subprocess routes (corresponding to a new individual) perform step b.

b) Apply a slightly different modified active or non delay schedule generation algorithm. The differences to be observed are the following:

- All the procedures concerning to selection of subprocess routes are not considered because the subprocess routes are already determined.
- In step 7 of the schedule generation algorithm, an operation $u^{(1)} \in S_i'$ must be selected to be scheduled next. This selection procedure is modified here to include the basic crossover operator. Thus, one must select to be scheduled next an operation $u^{(1)}$ such that $u^{(1)} \in S_i' \cap PS_{final(j)}$, and $u^{(1)}$ is the operation that first appears in $PS_{final(j)}$ among the operations from $S_i' \cap PS_{final(j)}$, for j=1 or j=2, where $PS_{final(j)}$ is the sorted operation set of parent j, i.e., the parent j chromosome. Algorithmically speaking, the selection of $u^{(1)}$ in step 7 of the modified schedule generation algorithm works as follows:
    1) Randomly select one of the two parents. Let j* be this parent. If $S_i' \cap PS_{final(j^*)} = \varnothing$ then the other parent must be assigned to j*. Note that this empty intersection can occur when all operations in S_i' belong to subprocess routes not used by parent j*.
    2) Select to be scheduled next the operation $u^{(1)} \in S_i' \cap PS_{final(j^*)}$, such that $u^{(1)}$ is the operation that first appears in $PS_{final(j^*)}$ among the operations in $S_i' \cap PS_{final(j^*)}$.
- In step 10 of the modified schedule generation algorithm, insert the operation $u^{(1)}$ in the sorted set of the offspring solution $PS_{final(offspring)}$

Mutation operators are applied to the offspring with low probability. As in the crossover, a subprocess route mutation and a basic mutation are implemented. The subprocess route mutation is implemented by randomly choosing a subprocess and changing its route to another randomly selected. The basic mutation chosen is the position based mutation, in which a randomly selected operation u* is arbitrarily moved from one position to another in PS_{final}. A position based mutation was adopted due to its superior results in classical JSSP (Mattfeld, 1996). The mutation operations are described below:

a) Subprocess route mutation: Randomly choose a subprocess and change its route to another randomly selected. In the sorted operation set PS_{final}, remove all operations of the old subprocess route and sequentially insert the operations of the new route in the same loci of the old ones. If the new route has more operations than the old route, the "exceeding" operations are inserted right after the old route last locus.

b) Basic mutation: Apply a position based mutation to the individual chromosome PS_{final}

c) Decoding: Apply a slightly different modified schedule generation algorithm. The differences to be observed are the following:
   - All the procedures concerning to selection of subprocess routes are not considered because the subprocess routes are already determined.
   - In step 7 of the MASGA, select to be scheduled next an operation $u^{(1)}$ such that $u^{(1)}$ is the operation that first appears in PS_{final} among the operations from $S_i' \cap PS_{final}$.
   - The operation $u^{(1)}$ must be inserted in the sorted set PS_{final_new} of the mutated solution in step 10 of the modified schedule generation algorithm.

Other GA applications dealing with alternative processing plans have been reported. As noted in chapter 3, the representation schemes used in such applications are very limited, being usually incomplete and redundant. Instead, the genetic algorithm proposed here can explore a large solution space and presents almost no redundancy. The solution search space is determined by the schedule generation strategy adopted. Comparisons between non delay and active generation strategies in the GA will be addressed in chapter 5. Also the proper set of algorithm parameters (e.g., crossover and mutation rates, population size) will be determined..

4.4.2. The Genetic Algorithm Framework

The final hybrid genetic algorithm framework work as follows, where random(1) is a random generated real number between zero and one:

1) i=0. Generate an initial population of good schedules P(0) using the modified active or non-delay schedule generation algorithms proposed in section 4.2.
2) For each individual of the population apply the local hill climbing algorithm described in section 4.3.
3) i = i + 1.
4) For j=1 to (population size)/2 do

4.1) select two parents within the population using a proportional selection scheme over scaled fitness values.

4.2) if random(1) < basic crossover rate

then

if random(1) < subprocess route crossover rate

then

generate two new individuals by applying the two crossover operators.

else

generate two new individuals by applying only the basic crossover operator

else

the two parents are assigned to the new two individuals without modification

4.3) for each of the two new individuals do

if random(1) < basic mutation rate then

apply the basic mutation to the new individual

if random(1) < subprocess route mutation rate then

apply the subprocess route mutation to the new individual

4.4) for each of the two new individuals do

if the new individual is not identical to one of its parents then

apply the local hill climbing to the new individual

4.5) include the two new individuals in population P(i)

5) If the best solution of P(i-1) is not already in P(i) then include it in P(i) in the place of the worst solution of P(i). That is, apply the elitism policy.

6) If i = maximum number of generations then stop, else return to step 3.

Note that the route crossover is applied only in conjunction with the basic crossover, but the mutation operators are independent of one another.

Genetic algorithms hybridized with local improvement procedures have been successfully used to minimize makespan in classical JSSP. In this chapter we extended this approach to deal with scheduling problems encountered in real production environments.

The framework proposed presents high modeling capabilities. It also supports (and requires) a number of heuristics that aid to guide the search process. A number of variations of the basic framework can be easily obtained . For instance, eliminating the hill climbing algorithm results in a " pure" GA approach. In the next chapter the influence of several heuristic and configuration options in the overall system performance is analyzed.

CHAPTER 5
EXPERIMENTAL RESULTS

5.1. INTRODUCTION

This chapter describes the set of experiments designed to determine the influence of several factors (configuration options) on the performance of the hybrid scheduling system.

A complex problem generator program was developed to aid simulation studies. The program is able to randomly generate problems with the large number of real world constraints considered here. The user specifies the probability distributions of 20 variables (e.g., setup time, time gap between consecutive maintenance breaks, number of parts directly preceding a part in the bill of material of a job, etc.), and also the values of 13 other variables and parameters (e.g., overlap policy, mean machine static loading rate, probability of occurrence of job and operation ready times, etc.), and the program randomly generates a problem instance. A set of 6 problems was used in the experiments. Basically, the problems vary in size, level of competition for resources and availability of alternative routes. The most important qualitative attributes of these benchmark problems are shown in the table 5.1 below. The problem generator program and the problems used here are detailed in appendixes A and B respectively.

Problem	Size (number of operations)	Level of competition for resources (other than machines)	Availability of alternative routes
1	Small (106)	Medium	Medium
2	Large (416)	Medium	Medium
3	Medium (237)	Medium	Medium
4	Small (142)	High	Medium
5	Small (116)	Low	Medium
6	Small (161)	Medium	High

Table 5.1. Description of the Problems

Note that the number of operations in the table above is the average number of operations really scheduled by the schedule generation algorithm. Because of the presence of alternative routes and machine maintenance operations the total number of operations in the problems is much higher. For instance, problem 1 has 151 operations and problem 6 has 323 operations.

The description of the problems given above suggests that the attributes size, level of competition for resources and availability of alternative routes should be used as factors in a factorial experiment. If the reader goes through appendix B, a different conclusion will be drawn. The problems considered are much too complex and it is impossible to fix one attribute and vary others because a large set of parameters are connected to each of these qualitative attributes. Therefore, experiments were conducted with each problem separately and results were put together to draw overall conclusions.

The performance of the proposed models is also related to the multiobjective function being used. The design of the ideal multiobjective function is a complex environment dependent problem and will not be addressed in this work. Although the system is prepared to deal with any combination of regular performance measures, all the experiments were conducted using the following function: F = Caver + Cmax = Mean completion time + Makespan. Due date dependent measures (e.g., tardiness, lateness, etc.) were not adopted in order to avoid the effect of due date tightness in the system performance. The influence of due date assignment method and due date tightness in the performance of different dispatching rules was discussed in chapter 2.

In the next three sections, the three basic modules of the system (the modified schedule generation algorithm, the local hill climbing and the genetic algorithm) are analyzed separately in order to understand the significance of several configuration options and adjust their levels. Afterwards, the entire hybrid system is studied.

All the experiments were conducted using a PC with a pentium 100 MHz processor and 32 MB of RAM memory.

5.2. MODIFIED SCHEDULE GENERATION ALGORITHMS

The modified schedule generation algorithm has a number of possible configurations. This section analyzes the effect of different configuration options under several problem instances. The simulations in this section are concerned only with the schedule generation algorithm. The objective here is to examine the effects of different program options on the quality of the initial solutions. The local hill climbing and the genetic algorithm will be addressed later .

A full factorial experiment was performed for each problem instance. The factors (configuration options) crossed in the experiment are described in the following table. The number of replicates was fixed at 200 (small mean variances are desirable for comparison purposes).

Factor (symbol)	Level	Description
Generation strategy (ACTNON)	1	Active schedule generation algorithm
	2	Non delay schedule generation algorithm
Route selection method (METROUT)	1	Random selection of subprocess routes
	2	Route selection heuristic
Dispatching procedure (DISP)	1	Random dispatching
	2	SPT (Shortest Processing Time rule)
Resource selection method (HEURITR)	1	Random selection of resources
	2	Minimum gap heuristic
Generation mode (GENMOD)	1	Complete generation mode
	2	Simplified generation mode

Table 5.2. Factors Crossed in a Factorial Experiment Related to the Modified Schedule Generation Algorithm

When all configuration options are set to level 1 the schedule generation algorithm is said to be in its *basic configuration*. Such configuration will be useful in next sections, where other system modules are tested. Note that basic configuration does not mean best configuration. It is only the one that allows the exploration of the largest search

space. Therefore, basic configuration adopts factor levels suitable to the exploration of a solution space as large as possible.

The value of *a* in the route selection heuristic was fixed at 0.75, where *a* is the relative importance of selecting fast subprocess routes and (1-a) is the relative importance of balancing resource utilization. This value worked well for several problem instances. In practice, however, this parameter must be adjusted to each particular environment.

Dispatching rules tend to confine the search to a small region of the solution spacem, causing premature convergence of the GA. Because of this only SPT dispatching rule was tested.

Table 5.3 shows the significant main effects and interactions for each problem and their respective P values. Only effects related to P values less than 0.075 are reported.

Problem	Significant effects - P values			
	Main effect	Two factors interaction	Three factors interaction	Four factors interaction
1	ACTNON - 0.000 METROUT - 0.000 DISP - 0.000 GENMOD - 0.002	ACTNON*DISP - 0.000 ACTNON*GENMOD - 0.000	ACTNON*DISP*GENMOD - 0.002	-
2	ACTNON - 0.000 METROUT - 0.000 DISP - 0.000 HEURITR - 0.006	ACTNON*METROUT - 0.053 ACTNON*DISP - 0.000 METROUT*DISP - 0.000 DISP*HEURITR - 0.000	METROUT*DISP*HEURITR - 0.013	-
3	ACTNON - 0.000 METROUT - 0.000 DISP - 0.000 GENMOD - 0.034	ACTNON*METROUT - 0.001 ACTNON*DISP - 0.000	ACTNON*METROUT*DISP - 0.000 ACTNON*METROUT*HEURITR - 0.044 METROUT*DISP*GENMOD - 0.018	ACTNON*METROUT*DISP*HEURITR - 0.024 METROUT*DISP*HEURITR*GENMOD - 0.030
4	ACTNON - 0.000 METROUT - 0.000 DISP - 0.000 HEURITR - 0.000	ACTNON*METROUT - 0.001 ACTNON*DISP - 0.000 ACTNON*HEURITR - 0.001 ACTNON*GENMOD - 0.001 METROUT*DISP - 0.004 DISP*HEURITR - 0.003 DISP*GENMOD - 0.000	ACTNON*DISP*GENMOD - 0.000	
5	ACTNON - 0.000 METROUT - 0.000	ACTNON*DISP - 0.000		
6	ACTNON - 0.000 METROUT - 0.000 DISP - 0.000 HEURITR - 0.007	ACTNON*DISP - 0.000 ACTNON*HEURITR - 0.004 ACTNON*GENMOD - 0.005 METROUT*DISP - 0.049 METROUT*HEURITR - 0.029 DISP*GENMOD - 0.005		ACTNON*DISP*HEURITR*GENMOD - 0.049

Table 5.3. Significant Effects - Modified Schedule Generation Algorithm

Variations in the evaluation function values due to changes in factor levels and the correspondent confidence interval for these differences (level of significant of 0.05) are reported in the next table. Because interactions are significant, these variation values are not due to changing main factor levels alone. Differences in CPU time are also shown in the table.

Problem	Significant main effects	Evaluation Function Value (Average)				Computational time - single run (sec)		
		factor at level 1	factor at level 2	confidence interval for differences	variation (%)	factor at level 1	factor at level 2	variation (%)
1	ACTNON	1741.599	1617.131	(119.6, 129.4)	7.1	0.022	0.021	4.5
	METROUT	1710.110	1648.620	(56.6, 66.4)	3.6	0.017	0.026	-52.9
	DISP	1700.335	1658.395	(37.0, 46.8)	2.5	0.023	0.021	8.7
	GENMOD	1675.493	1683.238	(-12.7, -2.8)	-0.5	0.022	0.021	4.5
2	ACTNON	4511.023	4268.095	(226.9, 258.9)	5.3	0.116	0.111	4.3
	METROUT	4511.285	4267.833	(227.5, 259.5)	5.4	0.095	0.132	-38.9
	DISP	4294.624	4484.494	(-205.9, -173.9)	-3.7	0.110	0.116	-5.5
	HEURITR	4400.848	4378.269	(5.7, 37.7)	0.5	0.113	0.113	0
3	ACTNON	4087.135	3746.413	(328.4, 353.0)	8.3	0.047	0.048	-2.1
	METROUT	3970.582	3862.966	(95.3, 119.9)	2.7	0.036	0.060	-66.7
	DISP	3826.746	4006.802	(-192.4, -167.8)	-4.7	0.048	0.048	0
	GENMOD	3910.129	3923.419	(-25.6, -1.0)	-0.3	0.048	0.047	2.1
4	ACTNON	2281.593	2073.377	(200.4, 216.0)	9.1	0.028	0.030	-7.1
	METROUT	2238.078	2116.892	(113.4, 129.0)	5.4	0.023	0.035	-52.2
	DISP	2106.700	2248.271	(-149.4, -133.8)	-6.7	0.028	0.030	-7.1
	HEURITR	2187.944	2167.027	(13.1, 28.7)	1.0	0.028	0.029	-3.6
5	ACTNON	1569.815	1483.535	(80.0, 92.6)	5.5	0.013	0.012	7.6
	METROUT	1539.040	1514.310	(18.4, 31.0)	1.6	0.008	0.015	-87.5
6	ACTNON	2751.061	2546.987	(194.9, 213.3)	7.4	0.033	0.032	3.0
	METROUT	2706.571	2591.477	(105.9, 124.3)	4.3	0.018	0.048	-166.7
	DISP	2636.495	2661.553	(-34.3, -15.9)	-1.0	0.033	0.033	0
	HEURITR	2655.351	2642.697	(3.5, 21.9)	0.5	0.033	0.033	0

Table 5.4. Experiment Results - Modified Schedule Generation Algorithm

The variation values above were calculated as follow: (Evaluation function value (or CPU time) with factor at level 1 - Evaluation function value (CPU time) with factor at level 2) / Evaluation function value (CPU time) with the factor at level 1. So positive values for the percentage variations mean enhancement in performance, since we are dealing with a minimization problem. All the evaluation function values in the table are averages over 16 treatments * 200 replicates = 3200 solutions.

The analysis showed a large number of significant interactions. An in-depth study of these interactions will not be conducted here. The interaction ACTNON*DISP, however, was highly significant in all problem instances and deserves special attention. A

behavior pattern can be drawn from the charts in figure 5.1 that graphically display this interaction in all problems. Clearly SPT rule is more effective in the non delay scheduling generation algorithm while random dispatching is more suitable for the active schedule generation algorithm.

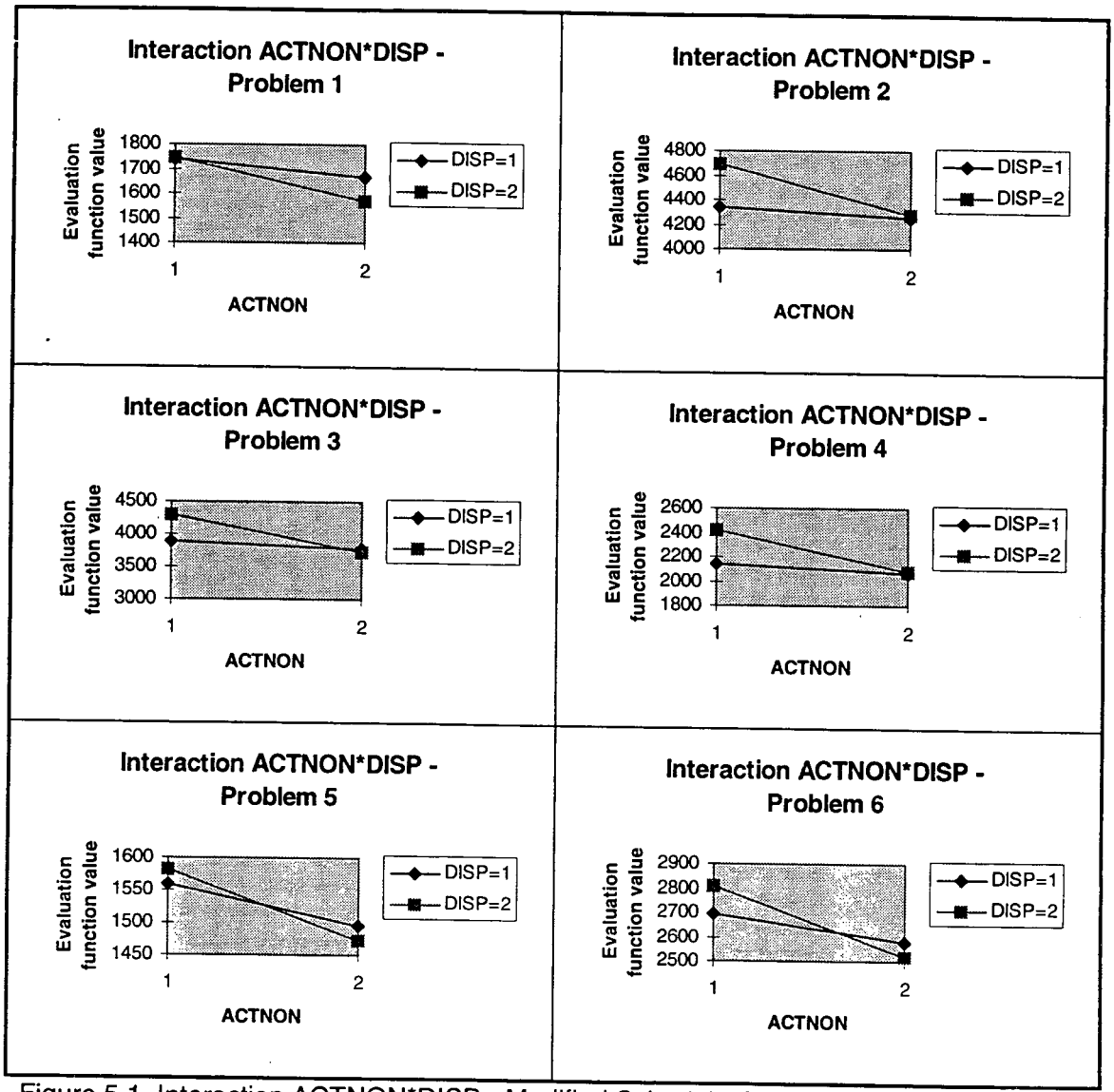


Figure 5.1. Interaction ACTNON*DISP - Modified Schedule Generation Algorithm

For each problem instance we stored the set of factor levels that produce the best results, and also the best set of factor levels with random dispatching procedure (DISP=1). This latter arrangement of best factor levels (with DISP=1) will be used later in the entire system experimental design. These two sets along with the basic configuration described earlier are shown in table 5.5.

Problem	Configuration	Factor levels					Average Evaluation Function Value	CPU time (sec)
		ACTNON	METROUT	DISP	HEURITR	GENMOD		
1	General best set of factor levels	2	2	2	2	2	1526.4	0.025
	Best set of factor levels with DISP=1	2	2	1	2	1	1628.2	0.025
	Basic configuration	1	1	1	1	1	1762.3	0.021
2	General best set of factor levels	2	2	1	2	1	4119.4	0.133
	Best set of factor levels with DISP=1	2	2	1	2	1	4119.4	0.133
	Basic configuration	1	1	1	1	1	4404.5	0.100
3	General best set of factor levels	2	2	2	2	1	3645.0	0.062
	Best set of factor levels with DISP=1	2	2	1	1	1	3686.9	0.067
	Basic configuration	1	1	1	1	1	3933.9	0.038
4	General best set of factor levels	2	2	1	1	2	2005.7	0.033
	Best set of factor levels with DISP=1	2	2	1	1	2	2005.7	0.033
	Basic configuration	1	1	1	1	1	2190.4	0.025
5	General best set of factor levels	2	2	2	2	2	1452.6	0.017
	Best set of factor levels with DISP=1	2	2	1	1	1	1478.3	0.017
	Basic configuration	1	1	1	1	1	1569.0	0.012
6	General best set of factor levels	2	2	2	1	2	2443.0	0.046
	Best set of factor levels with DISP=1	2	2	1	1	2	2517.9	0.046
	Basic configuration	1	1	1	1	1	2736.1	0.021

Table 5.5. Arrangements of Factor Levels - Modified Schedule Generation Algorithm

For all problems the non delay schedule generation algorithm performed on average 7.1% better than the active one. The route selection heuristic worked well in all problem instances, enhancing the average system performance by 3.8%, but also increased the computational time by 77%. In problem 6, characterized by large number of alternative routes, the increase in CPU time due to the route selection heuristic reached 166%. When the non delay schedule generation strategy was being used, SPT rule achieved an average performance 1.5% higher than random dispatching. However, SPT rule degraded the average system performance by 6.2% when the active schedule generation strategy was adopted. The main effect of factor generation mode (GENMOD) was determined to be significant only in problems 1 and 3, where using the complete generation mode improved the system performance by about 0.4% at a small computational cost. The resource selection method presented significant effect in 3 out of the 6 problems. In such problems, the use of the minimum gap heuristic causes an average gain in performance of 0.7% compared to random selection of resources.

5.3. LOCAL HILL CLIMBING

This section investigates the performance of the local hill climbing algorithm and its configuration options. Here, the local search algorithm is examined alone. Its effect in the entire hybrid genetic system will be addressed in section 5.4.

Similar to the previous section, for each of the six problem instances a full factorial experiment with 200 replicates for treatment was performed. All the initial solutions were generated by the scheduling generation algorithm in its basic configuration (ACTNON = METROUT = DISP = HEURITR = GENMOD = 1). A description of the factors crossed in this experiment is given in the following table:

Factor (symbol)	Levels	Description
Resource changing between moves (RESCH)	1	Resources (other than machines) required to execute an operation **are allowed** to change when a move in the neighborhood occurs.
	2	Resources required to execute an operation are not allowed to change when a move is performed
Method for selecting the arc to be reversed (ARCTYPE)	1	Bottleneck heuristic
	2	Random choice of the arc to be reversed

Table 5.6. Factors Crossed in a Factorial Experiment Related to the Local Hill Climbing

Note that the selection of the arc to be reversed is made among the candidate arcs in the neighborhood structure defined in last chapter, no matter which selection method is being adopted (random or bottleneck heuristic).

The improvement in the evaluation function values achieved by the local hill climbing over the initial solutions generated by the schedule generation algorithm (in its basic configuration) is reported in the following table. The computational time of the local hill climbing is also described.

Problem	Average Evaluation Function Value			Computational time - single run (sec)	
	Initial solution	Initial solution enhanced by the local hill climbing	Variation (%)	Initial solution	Initial Solution enhanced by the local hill climbing
1	1759.2	1719.0	2.3	0.021	0.032
2	4438.2	4414.9	0.5	0.100	0.235
3	3943.8	3890.3	1.4	0.038	0.100
4	2184.5	2171.8	0.6	0.025	0.044
5	1580.6	1546.6	2.2	0.012	0.023
6	2725.0	2698.1	1.0	0.021	0.041
Average:			1.3		

Table 5.7. Solution improvement Due to the Local Hill Climbing Procedure

The evaluation function values in the local hill climbing column are averages over all treatments for which RESCH=1, that is, averages over 2 treatments * 200 replicates = 400 values. Variations in the evaluation function and CPU time were calculated as described in last section for the schedule generation algorithm.

Table 5.8 shows the results of the full factorial experiment. Only effects related to P values less than 0.075 are reported. In problems 2, 4 and 5 no significant main effect or interaction were determined.

Problem	Significant main effects and interactions	P value	Average Evaluation Function Value				Computational time - single run (sec)		
			factor at level 1	factor at level 2	confidence interval for differences	variation (%)	factor at level 1	factor at level 2	variation (%)
1	RESCH	0.021	1719.0	1736.9	(-33.1, -2.7)	-1.0	0.032	0.022	31.2
3	RESCH	0.031	3890.3	3921.7	(-59.7, -3.1)	-0.8	0.100	0.061	39.0
6	ARCTYPE RESCH*ARCTYPE	0.070 0.054	2693.7	2716.6	(-47.6, 1.8)	-0.9	0.036	0.033	8.3

Table 5.8. Experiment Results - Local Hill Climbing

The weak interaction between RESCH and ARCTYPE (P value of 0.059) observed in problem 6 is graphically displayed below. As it occurred in only one case no general conclusions can be made.

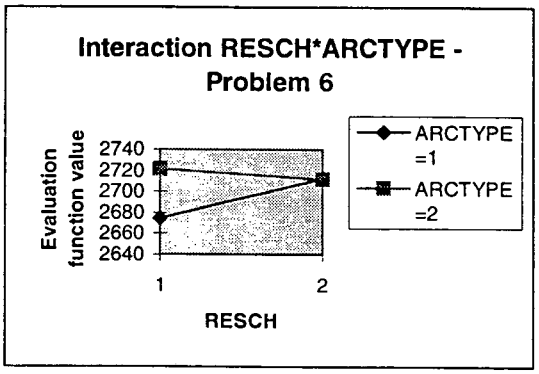


Figure 5.2. Interaction RESCH*ARCTYPE - Local Hill Climbing

For each problem instance we also stored the set of factor levels that produced the best results. A factor was set to its default level when its main effect was not significant and the factor was not present in any significant interaction. The default levels for RESCH and ARCTYPE are 1 and 1 respectively. When both RESCH and ARCTYPE are adjusted to their default levels, the local hill climbing is said to be in its basic

configuration. For each problem instance the following arrangements of factor levels provided the best results:

Problem	1	2	3	4	5	6
RESCH	1	1	1	1	1	1
ARCTYPE	1	1	1	1	1	1

Table 5.9. Arrangements of Factor Levels - Local Hill Climbing

Note that for all problems the basic and best configurations coincided for all problems. From now on this basic configuration of the local hill climbing will be used in all problems.

Some important conclusions may be drawn from the experimental analysis. The local hill climbing procedure is more efficient in low constrained problems because the representational scheme and the neighborhood structure do not directly take into account resources other than machines. The availability of these resources is considered only as constraints to be satisfied. The moves are performed by reversing arcs connecting operations executed on the same **machine.** The local search procedure also worked better in smaller problems. Allowing resources other than machines to change when moves are implemented significantly enhanced the performance of the hill climbing procedure in problems 1 and 3, but also required greater CPU time as shown in table 5.8. No significant difference in performance between the methods for selecting the arc to be reversed was observed. Only in problem 6 a P value of 0.07 suggests an improvement in performance due to the bottleneck heuristic.

5.4. GENETIC ALGORITHM PARAMETERS

In this section values of parameters associated with the genetic algorithm are determined. Specifically, we want to find out proper crossover and mutation rates, population size and number of generations. Such parameter values will be used in the entire system simulation carried out in next section. Only problem 1 was used in the experiments, since these GA parameters are very robust. The GA used the active solution generation strategy, and basic configurations were adopted in both schedule generation and local hill climbing algorithms.

Crossover and mutation rate:

The basic crossover and mutation rates were determined through a factorial experiment (with 10 replicates per treatment) in which the level of factors were the following:

basic crossover rate:	0.6	0.8	1.0	
basic mutation rate:	0.01	0.03	0.05	0.12

Table 5.10. Factors Crossed in a Factorial Experiment to Select GA Parameter Values

Population size and number of generations were fixed at 100. As shown in the ANOVA table below, the main effects of crossover and mutation were determined to be significant at a level of significance of 0.01. Surprisingly, the interaction was not significant, even for a level of 0.05. The values adopted to basic crossover rate and basic mutation rate were 0.8 and 0.05 respectively.

Source	Degrees of freedom	Sum of squares	Mean square	F	F0.05	F0.01
crossover	2	228.47	114.23	6.37	3.10	4.82
mutation	3	263.62	87.87	4.90	2.70	4.01
interaction	6	164.92	27.49	1.53	2.20	2.98
error	99	1775.13	17.93			

Table 5.11. ANOVA Table - Genetic Algorithm Operators

The importance of route crossover and route mutation depend on the problem under consideration. Some experiments showed that their effects can be disregarded in problems with low level of alternative routes. For problems with medium to high level of alternative routes, values of 0.6 for route crossover rate and 0.1 for mutation rate showed to be appropriate.

Population size:

Larger population sizes avoid premature convergence and tend to produce better solutions. However, there is a saturation of this tendency.

In order to determine a suitable population size and confirm the saturation effect described above, our genetic algorithm was run with several population sizes. The chart below shows the experimental results, where the best fitness (evaluation function value) at each population size is an average over 20 runs of the genetic algorithm.

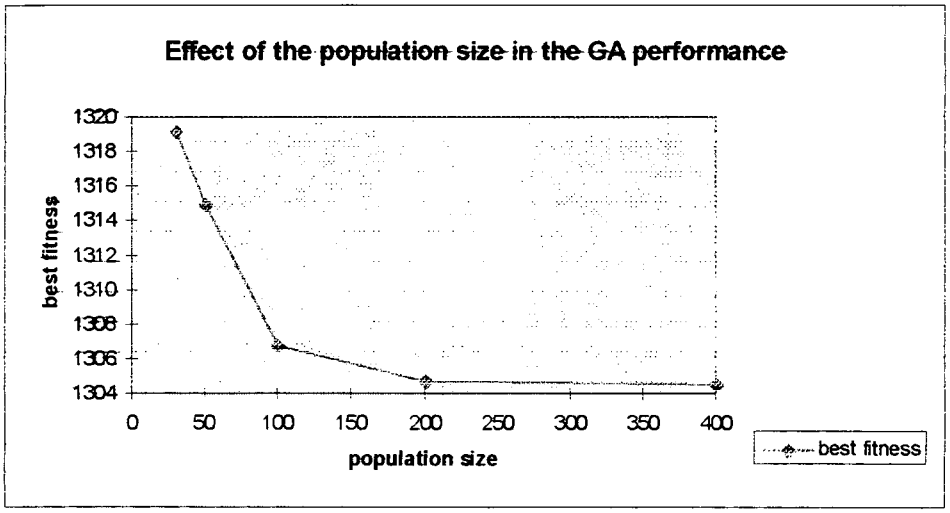


Figure 5.3. Effect of Population Size in the GA Performance

The number of generations was fixed at 100. Therefore, the number of trials is the population size times 100. Crossover and mutation rates were the ones determined earlier in this section.

Number of generations:

The termination criterion used in the GA algorithm proposed here is simply a fixed number of generations. In order to determine the effect of the number of generations in the genetic algorithm performance, 20 runs of the GA using 400 generations were carried out. The chart below shows the results obtained. The fitness values are averages over the 20 runs.

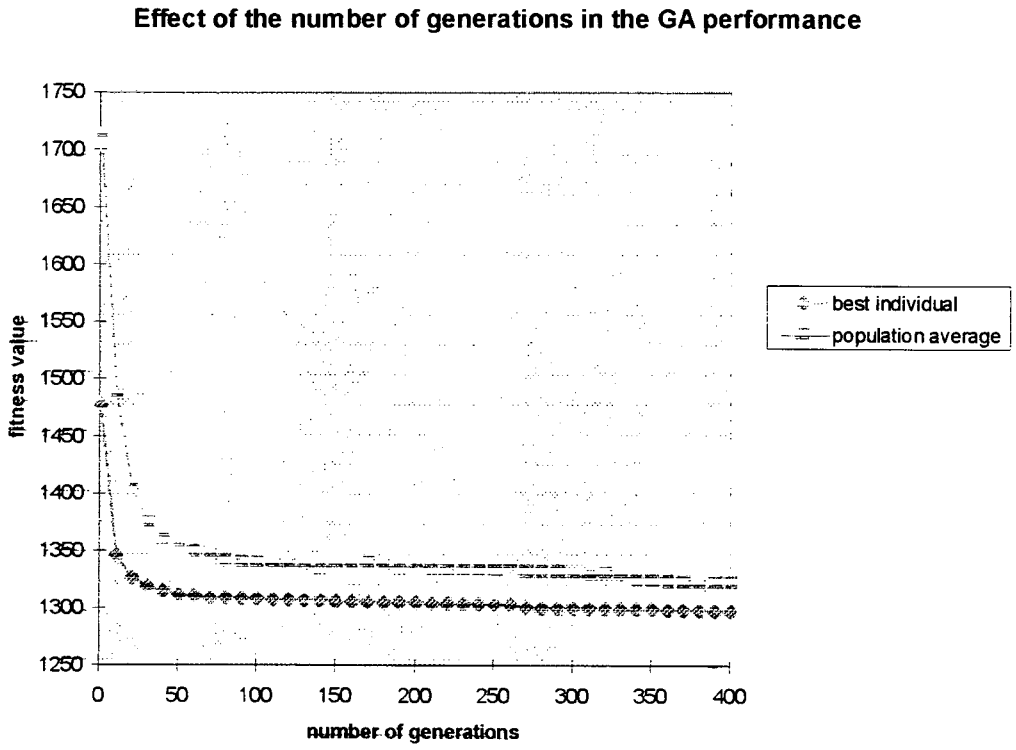


Figure 5.4. Effect of Number of Generations in the GA Performance

Population size was fixed at 100, and, therefore, the number of trials is the number of generations times 100. Crossover and mutation rates were set to the values determined earlier.

Genetic Algorithm Basic Configuration:

Based on the experiments described in this section and considering computational time also a performance measure, the following set of parameters was chosen to define the "basic configuration" of the GA proposed:

- basic crossover rate: 0.8
- basic mutation rate: 0.05
- route crossover rate: 0.6
- route mutation rate: 0.1
- population size: 100
- number of generations: 100

Note that larger population sizes and number of generations can improve the performance of the algorithm. Considering problem 1 as an example, the best fitness obtained with a population size of 400 is only about 0.2% greater than the one achieved by the basic configuration. Similarly, the use of 400 generations increases the average GA performance by only 0.8%. However, the increase in computation time reaches 400%. Therefore, the use of larger number of generations and population sizes is justified only in cases where enough computation time is available.

5.5. THE HYBRID GENETIC SYSTEM

This sections analyzes the performance of the entire system over the set of problem instances already mentioned in previous sections.

The configuration options are now considered in aggregated levels. The influence of the following factors in the overall system performance was studied in a full factorial experiment:

Factor (symbol)	Levels	Description
Quality of initial solutions (INIT)	1	Initial solutions generated by the basic configuration of the schedule generation algorithm.
	2	Schedule generation algorithm in its best configuration (with DISP=1) for each problem.
Hybridization level (CONFOP)	1	Genetic algorithm hybridized with a local hill climbing procedure.
	2	Pure GA
GA generation strategy (GACTNON)	1	Active
	2	Non delay

Table 5.12. Factors Crossed in a Factorial Experiment of the Entire Hybrid System

When INIT is in level 2 a set of good initial solutions are produced, and when INIT is in level 1 a set of distant initial solutions are produced. That is, level one drives the initial solutions to promising but small regions of the search space, while level 2 spread the initial population of solutions in the whole search space.

We observed that SPT rule (DISP=2) always lead to a premature convergence of the GA. To avoid this, we selected the best configuration of the schedule generation algorithm for each problem with the requirement of DISP=1.

Each time reproduction occurs in the GA, a schedule generation algorithm is used to create the new individual. As described in chapter 4, the crossover operator is embedded in the schedule generation algorithm. The GACTNON factor analyzed here is pertaining to the use of active or non delay generation strategy in the GA. The adoption of an active or non delay strategy during the generation of the initial solutions is a different issue. In the experiment the genetic algorithm and the local hill climbing were set to their basic configurations. The number of replicates per treatment was fixed at ten.

At first, differences in fitness between the first and last generations are reported. The following table shows the fitness of the best individual in the first and last generations as well as the average population fitness in the first and last generations. The fitness values in the table are averages over all treatments, that is averages over 8 treatments * 10 replicates = 80 values.

Problem	Best individual			Population Average			Computational
	Generation		Variation (%)	Generation		Variation (%)	Time - single run
	first	last		first	last		(min.)
1	1460.5	1312.9	10.1	1684.1	1338.1	20.5	3.5
2	3752.6	3493.2	6.9	4286.0	3547.5	17.2	19.8
3	3387.8	3076.2	9.2	3797.5	3113.8	18.0	9.6
4	1782.3	1624.6	8.8	2096.0	1685.2	19.6	4.5
5	1285.8	1222.6	4.9	1518.9	1241.2	18.3	1.8
6	2242.2	1932.4	13.8	2624.4	2025.3	22.8	3.4

Table 5.13. Average System Performance

The following charts show the average behavior of the system through time:

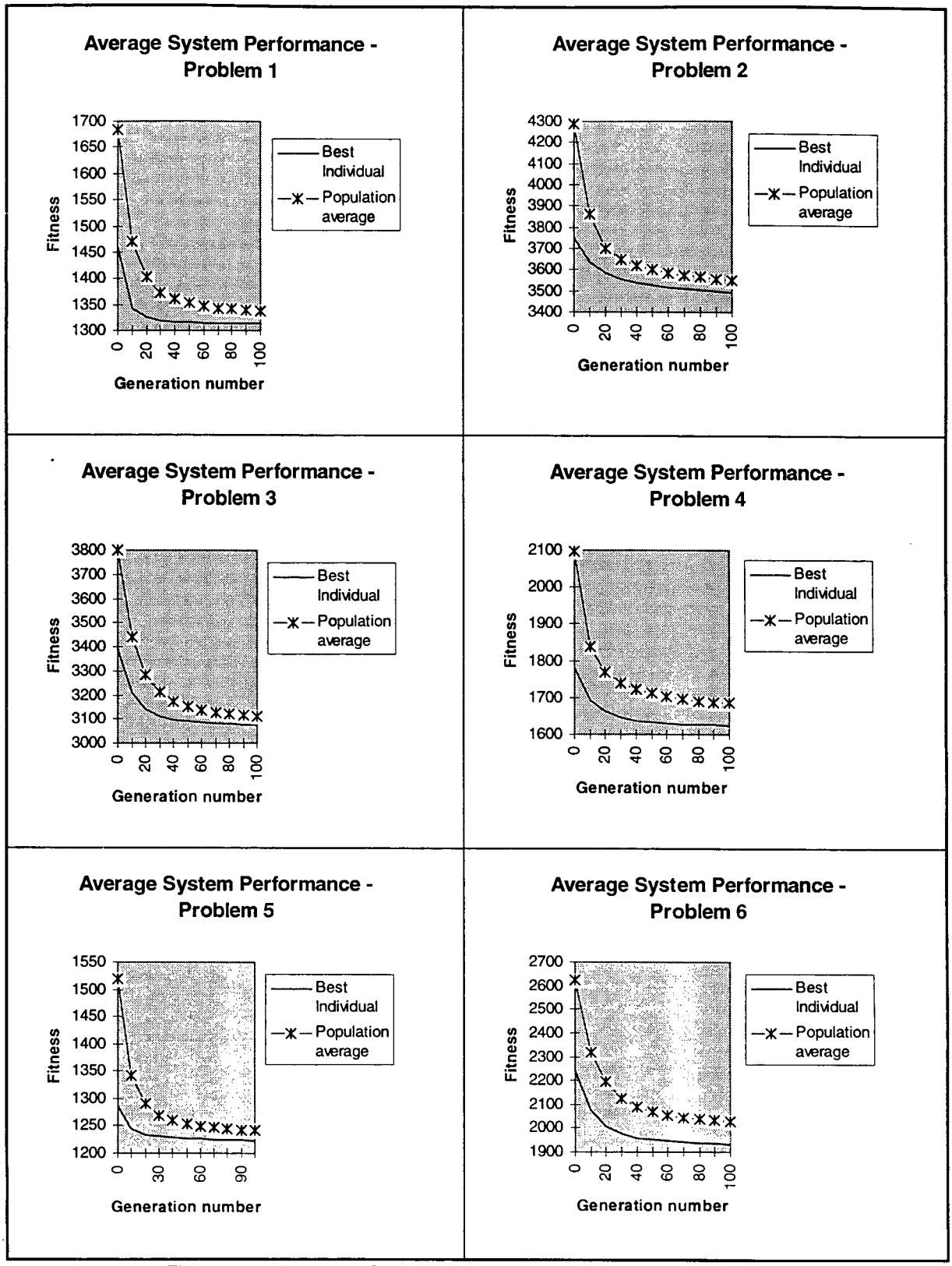


Figure 5.5. Average System Performance through Time

The next table shows the significant main effects and interactions for each problem and their respective P values. Only effects related to P values less than 0.075 are reported. Fitness variations due to changes in factor levels and the correspondent confidence interval for these differences (level of significant of 0.05) are also displayed. Note that when there is a significant interaction involving a factor, the variation is not due to changing in the level of the factor alone. Computational times are also addressed.

Problem	Significant main effects and interactions	P value	Fitness of the best individual in the last generation				Computational time - single run (min)		
			factor at level 1	factor at level 2	confidence interval for differences	variation (%)	factor at level 1	factor at level 2	variation (%)
1	GACTNON	0.000	1319.9	1305.9	(11.7, 16.3)	1.1	3.6	3.3	8.3
	INIT	0.000	1309.2	1316.6	(-9.7, -5.1)	-0.6	3.6	3.3	8.3
	CONFOP	0.000	1307.6	1318.2	(-12.9, -8.3)	-0.8	4.5	2.4	45.8
	GACTNON*INIT	0.000							
	GACTNON*CONFOP	0.000							
2	GACTNON	0.000	3510.4	3475.3	(23.9, 46.3)	1.0	19.9	19.7	1.0
	INIT	0.059	3498.3	3487.4	(-0.3, 22.1)	0.3	20.5	19.1	6.8
	CONFOP	0.000	3473.8	3511.9	(-49.3, -26.9)	-1.1	25.5	14.1	44.7
	GACTNON*INIT	0.007							
3	CONFOP	0.000	3057.3	3095.0	(-47.8, -27.6)	-1.2	13.6	5.6	58.8
4	GACTNON	0.000	1648.2	1600.9	(39.1, 55.5)	2.9	4.6	4.4	4.3
	INIT	0.000	1638.2	1611.0	(19.0, 35.4)	1.6	4.7	4.3	8.5
	CONFOP	0.026	1619.8	1629.3	(-17.7, -1.3)	-0.6	5.7	3.3	42.1
	GACTNON*INIT	0.005							
5	GACTNON	0.000	1215.1	1230.1	(8.1, 21.9)	-1.2	1.9	1.8	5.3
6	GACTNON	0.000	1950.8	1914.1	(26.3, 47.1)	1.9	3.5	3.2	8.6

Table 5.14. Experiment results - Hybrid Genetic System

Problems 1, 2 and 4 presented significant interactions between factors GACTNON and INIT and problem 1 also presented a significant interaction between GACTNON and CONFOP . These interactions are graphically described in figure 5.6, where "fitness" concerns to the evaluation function value of the best solution in the last generation.

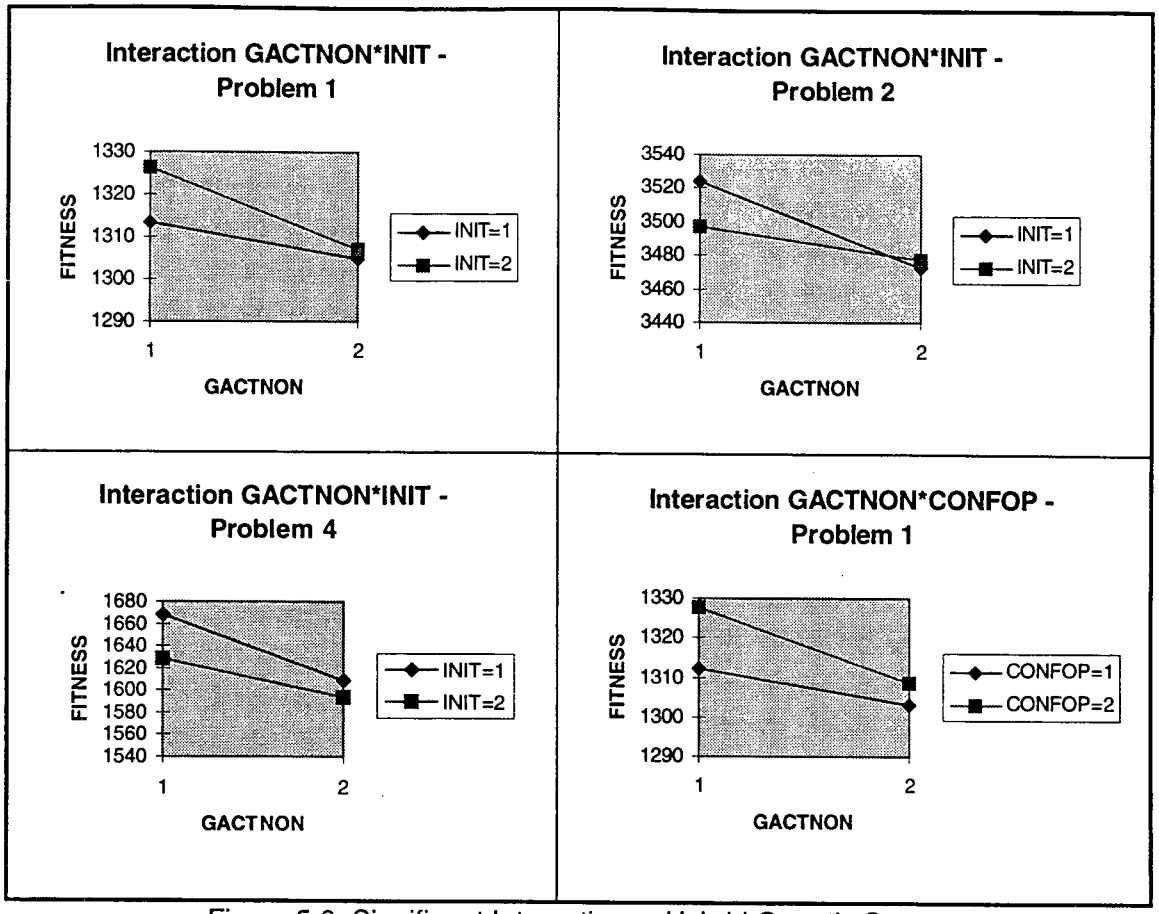


Figure 5.6. Significant Interactions - Hybrid Genetic System

Charts in figure 5.7 show the differences in convergence properties due to variations in the generation strategy adopted by the GA. The points plotted on the charts are averages over all treatments that present the same level of GACTNON, that is, each point on the charts is an average over 4 treatments * 10 replicates = 40 values. As the factor GACTNON is encountered in several significant interactions, the differences in behavior between the curves of GACTNON=1 and GACTNON=2 can not be attributed to differences in GACTNON levels alone.

Similarly, the charts in figure 5.8 compare the evolution of the population fitness when INIT=1 (initial solutions created by the basic configuration of the schedule generation algorithm) and INIT=2 (initial population created by the schedule generation algorithm in its best configuration, given that DISP=1) for all problems.

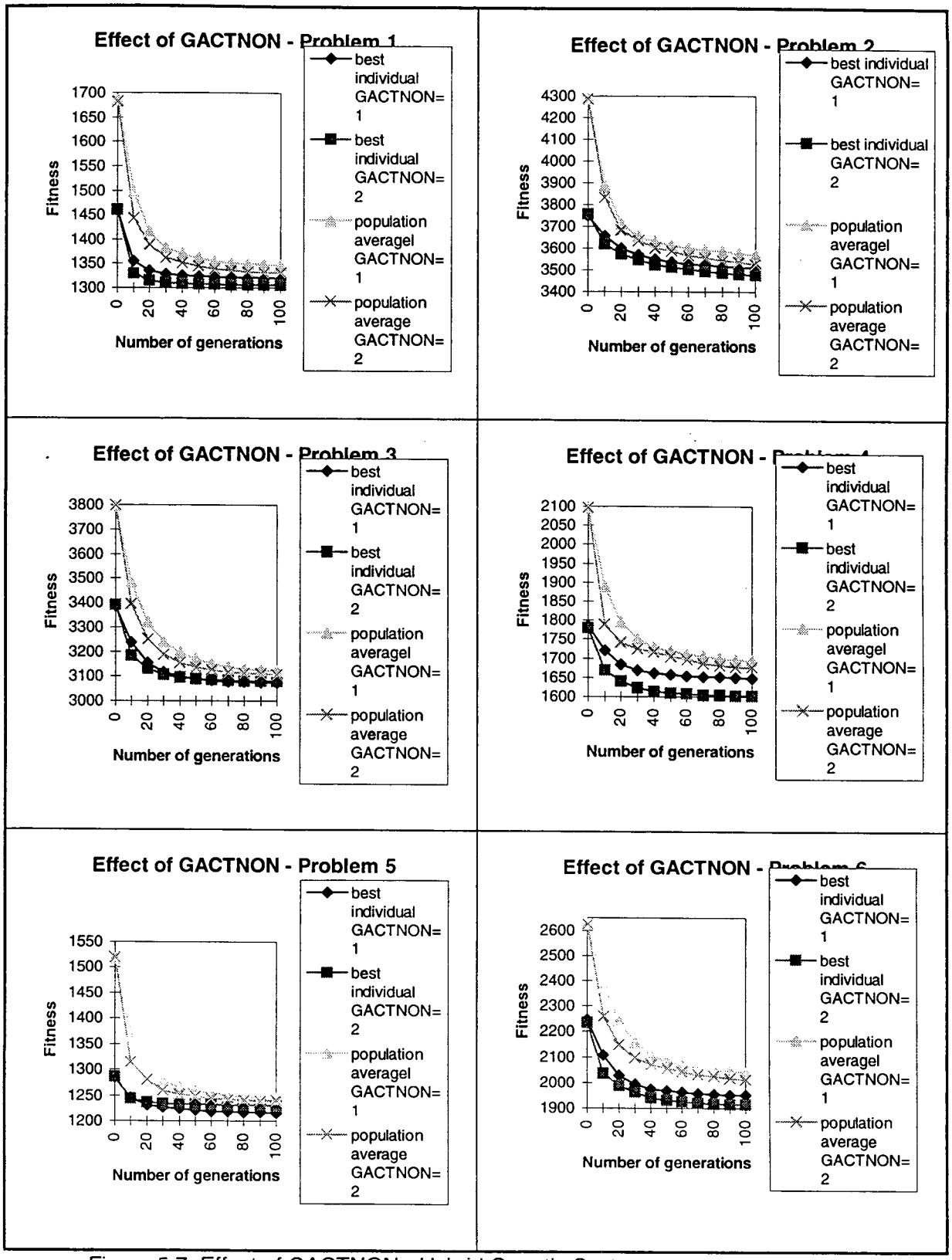


Figure 5.7. Effect of GACTNON - Hybrid Genetic System

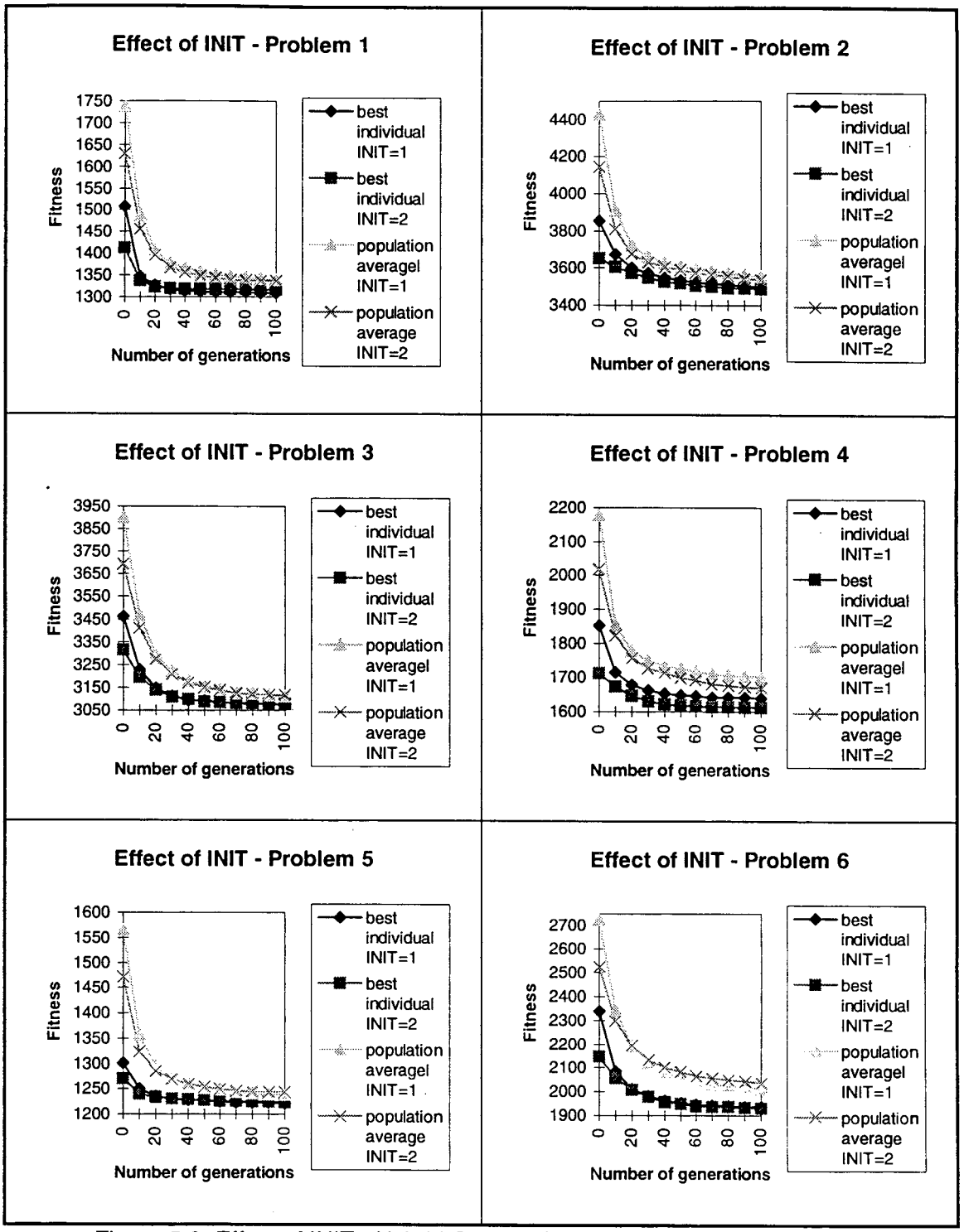


Figure 5.8. Effect of INIT - Hybrid Genetic System

Finally, the charts comparing the pure GA and the GA hybridized with a local hill climbing are plotted in figure 5.9.

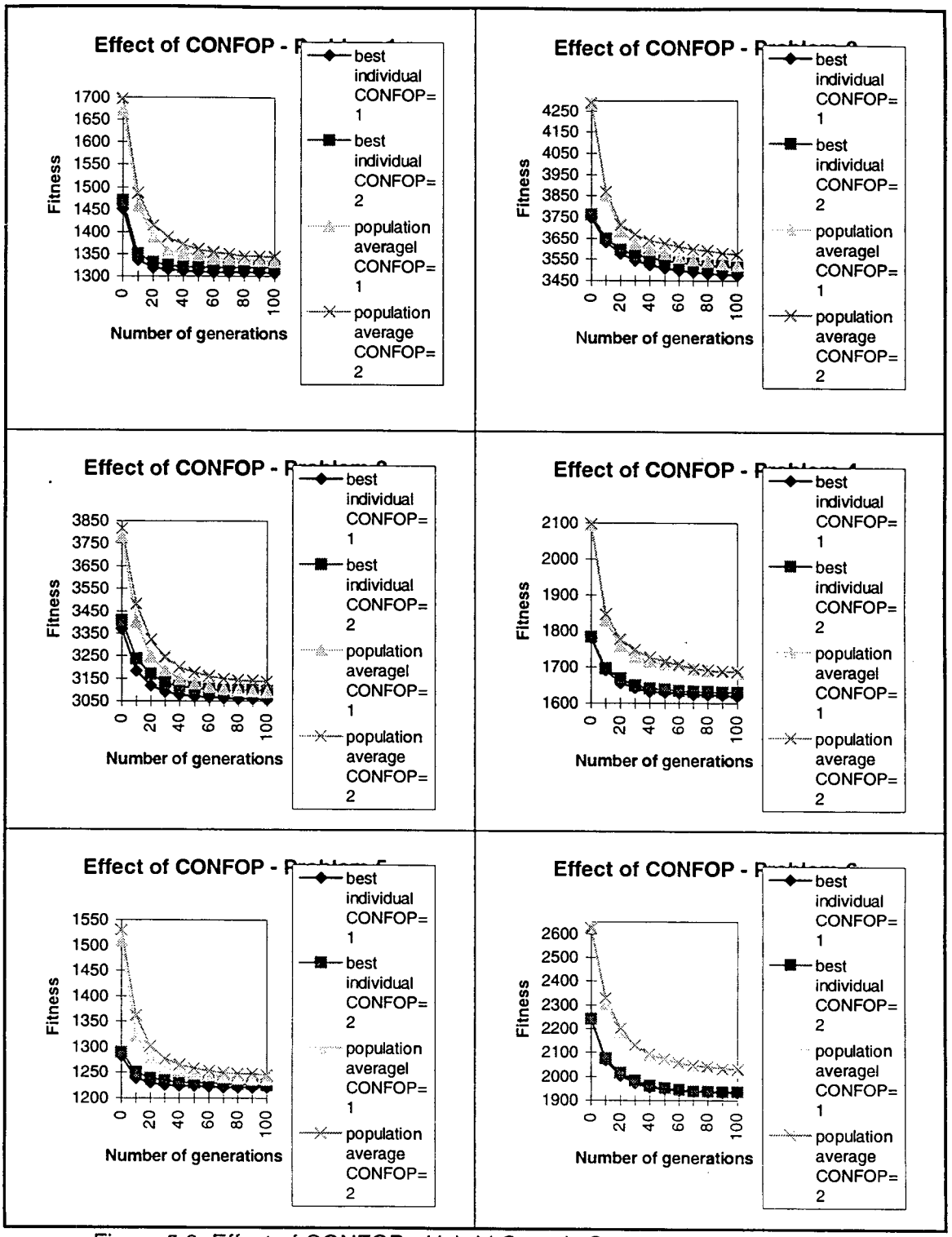


Figure 5.9. Effect of CONFOP - Hybrid Genetic System

Based on the tables and charts printed above a number of general conclusions can be drawn.

The generation strategy adopted by the GA (GACTNON) showed significant main effects in 5 out of 6 problems. Except for problem 5, the non delay strategy significantly achieved better results than the active one.

The main effect of factor INIT (quality of initial solutions) was significant in only 3 problems (including problem 2 where the P value was 0.059). In problems 2 and 4 the use of a better set of initial solutions (INIT=2) showed positive contributions to the system performance, and in problem 1 the better set of initial solutions led to a premature convergence of the GA, degrading the system performance. In all problems where INIT was significant, its interaction with GACTNON was significant either. The interaction charts suggested that the level of INIT is not important if the non delay generation strategy is adopted by the GA. This arrangement, where INIT=1 or 2 and GACTNON=2, seems to be the best for all problems.

The factor CONFOP (hybridization level) presented significant main effect in 4 out of 6 problem instances. The hybridization of the GA with the local hill climbing enhanced the system performance in all problems instances where CONFOP is significant. Even in problem 4, which is characterized by high competition level for resources other than machines, the hybridization significantly improved the system performance. Nevertheless, this hybridization is time expensive, increasing the CPU time required by about 47%.

The "Effect of GACTNON" charts showed a slightly sooner convergence of the system when the GA is using a non delay generation strategy (GACTNON=2). However, this convergence difference is not enough to make active generation a better strategy. Non delay generation strategy presented an overwhelming superiority over the active generation strategy in 5 out of 6 problems. Similarly the "Effect of INIT" charts showed a trend to a more premature convergence of the system when INIT=2. No significant difference in convergence rate was observed between CONFOP=1 and CONFOP=2.

The problem size, the competition level for resources and the hybridization with the local hill climbing procedure are crucial factors in the determination of the computational time required.

5.6. REMARKS

The proposed system worked well in all problem instances, showing to be a promising tool to solve real make to order job shop scheduling problems. The following guidelines aid the proper use of the system.

The generation strategy in the GA must be always the non delay one. If the non delay strategy is adopted by the GA the use or not of heuristics to improve the quality of initial solutions is not significant. However, these heuristics must be adopted if CPU time is scarce, once they lead the system to a premature convergence, and thus allowing a smaller number of generations in the GA. The hybridization with a local hill climbing procedure is always desirable unless there is not enough computational time.

An important question that has not been answered so far is if the proposed scheduling system efficiently solves classical job shop scheduling problems. When all real world constraints and alternatives described in chapter 4 are excluded from the production environment the resulting problem is a classical JSSP. Let the input problem be a classical JSSP. If the local hill climbing procedure is disabled then the scheduling system presented in this thesis will be identical to the GA with Active Schedule Constructive Crossover proposed by Park and Park (1995) to solve classical JSSPs. Park's genetic algorithm yielded outstanding results on five benchmark problems: ABZ6, CAR4, LA22, MT10 and MT20. We used four Taillard's benchmark problems to verify the performance of the proposed model on classical job shop scheduling problems. Taillard's benchmarks are characterized by their large size. The best hybrid genetic algorithm used so far to deal with these benchmarks is the GA3 proposed by Mattfeld (1996). The results obtained by our GA are comparable (although slightly worse) to the obtained by Mattfeld. These results are reported in Appendix D.

CHAPTER 6
CONCLUSIONS AND FURTHER RESEARCH

6.1. THESIS ORIGINALITY AND CONTRIBUTION

This work undertook the application of constructive heuristics, local search procedures and genetic algorithms to real make-to-order job shop scheduling problems.

To our knowledge, the first local search and genetic based system able to simultaneously consider all the features described in chapter 4 was implemented in this thesis. The consideration of these features by the scheduling system is crucial in the majority of real job shop production environments. At a micro level, the thesis is original due to the following four basic contributions:

1. Efficient active and non-delay schedule generation algorithms that consider jobs with bill of materials, alternative processing plans for parts and alternative resources for operations, requirement of multiple resources to process an operation, resource calendars, batch overlap, operation and job ready times, and sequence dependent setups were developed. These constructive algorithms were used to generate initial solutions in the hybrid scheduling system proposed here. Also, some heuristics (route selection and minimum gap heuristics) were proposed to improve the performance of the schedule generation algorithms.

2. The graph representation scheme (used so far in classical JSSPs) was extended to support jobs with bill of materials, alternative resources (except machines) for operations, requirement of multiple resources to process an operation, resource calendars, batch overlap, operation and job ready times, and sequence dependent setups. A neighborhood structure to be applied to the extended graph formulation was created and validated. The proposed neighborhood also supports multiobjective evaluation functions. Any local search strategy (e.g., simulated annealing and tabu search) can be easily implemented using the extended graph formulation and neighborhood structure proposed. In the system proposed here only a local hill climbing was implemented because the search diversification is provided by the GA.

3. A robust, complete and almost non-redundant genetic algorithm to solve real job shop scheduling problems that directly consider in the search process all the real world features (taken into account by the modified schedule generation algorithms) was presented. The proposed GA also supports multiobjective evaluation functions.

4. The modified schedule generation algorithm, the local search procedure and the genetic algorithm were hybridized to create a high performance scheduling system. The system was shown to generate high quality solutions for scheduling problems with large number of real world constraints and alternatives. The system efficiency and efficacy was observed in several different problems, i.e., the system performed well in problems of different sizes, different levels of competition for resources, different availability of alternative routes, etc.

Summarizing, we made possible high performance techniques (used so far to solve classical JSSPs) to be applied to complex real JSSPs.

6.2. SOME MODELING REMARKS

There are other real world constraints that have not been directly considered by the proposed system. The most important one is related to transportation times and material handling devices. If transportation devices are always available (infinite material handling capacity) only slight modifications in the overlap time equations are required to take transportation times into account. Let tr_u be the time required to transport a minimum transport batch of operation u (α_u) from the machine cell where operation u was processed to the machine cell where suc(u) will be processed. The overlap time to_u under policy 1 is now calculated as follows:

a) $to_u = ts_u + \alpha_u * top_u + tr_u$, if $t_{suc(u)} > \alpha_u * top_{suc(u)} + (n - \alpha_u) * top_u - tr_u$

b) $to_u = t_u - t_{suc(u)} + \alpha_u * top_{suc(u)}$, if $t_{suc(u)} \leq \alpha_u * top_{suc(u)} + (n - \alpha_u) * top_u - tr_u$

Equations for to_u under overlap policy 2 can be obtained in the same manner.

When there are constraints about availability of transportation devices (finite material handling capacity) the average time a transportation device takes to be available after it has been requested must be added to tr_u, i.e., tr_u = transportation time of α_u + average time to the requested transportation device be available. This approach is only reasonable for production environments where the ratios tr_u/t_u are small.

Up to this point the dynamic nature of the shop has not been addressed. The author advises the production environment to be rescheduled when the problem data significantly change. That is, the system must run again with updated data if rush orders arrive or non expected machine breakdowns, staff missing, changes in job specifications, etc. occur during the schedule execution.

6.3. FURTHER RESEARCH

Extensions of this work are recommended in three different directions:

1. ***Improvement of system performance***: Different representation schemes, neighborhood structures, local search control strategies, genetic operators and population management strategies may be proposed in order to achieve higher performance. The hybridization of the GA with more complex local search strategies like tabu search is also a promising research area.

2. ***Additional modeling capabilities***: Other real world constraints can be included in the scheduling system. For instance, buffer size limitations, operation waiting time constraints, batch splitting and grouping, batch preemption, and constraints related to availability of material handling devices are some important real world constraints not considered by the system proposed. A robust rescheduling methodology able to efficiently reschedule the job shop while keeping a high similarity with the previous schedule would also be very useful to avoid the chaos that usually comes with the rescheduling activity.

3. ***Multiobjective function design:*** The proposed system supports any multicriteria regular performance measure. The design of the multiobjective function however is beyond the scope of this work. How to determine the weight of each single criterion in

the multicriteria function and which criteria must be included are important questions that arise in real production environments. Neural network and inductive learning algorithms (e.g., C4.5 algorithm due to Quinlan, 1990) can be used to overcome this problem. Given a set of features describing the system status (e.g., priorities of job orders, expected mean machine loading rate) and a score for each single criterion (provided by the PPC manager) the "smart" algorithm would yield the proper multiobjective function to be used. The training phase would be "manager-specific."

Finally the robustness of the system proposed must be confirmed in real manufacturing enterprises.

REFERENCES

Aarts, E.H., Laarhoven, P.J.M., Lenstra, J.K., Ulder, N.L.J., "A computatioan study of local search algorithms for job shop scheduling," ORSA Journal on Computing, vol.6, n.2, 118-125, 1994.

Adams, J., Balas, E. and Zawack, D., "The shifting bottleneck procedure for job shop scheduling," Management Science, vol. 34, n. 3, pp. 391-401, 1988..

Adshead, N. S. and Price, D. H. R., "Adaptative scheduling: the use of dynamic due dates to accomodate demand variance in a make-for-stock shop," Int. J. Prod. Res., v. 26, n,7, pp. 1241-1258, 1988.

Ahiro, S. S., Isoda, K & Awane, H, "Input Scheduling and load balance control for a job shop," Int. J. Prod. Res., vol.22, n.4, 597-605, 1984.

Ahmad, I. and Dhodhi, M. K., "Multiprocessor Scheduling Using a Problem-Space Genetic Algorithm", IEE Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, pp.152-157, 1995.

Alidaee, B., "Schelule of n jobs on two identical machines to minimize weighted mean flow time," Comp. ind. Engng, vol.24, n.1, pp. 53-55, 1993.

Amar, A. D. and Gupta, J. N. D., "Simulated Versus Real Life Data in Testing the Efficiency of Schedulig Algorithms," IIE Transactions, pp. 16-25, March 1986.

Applegate, D and Cook, W., "A Computational Study of the Job-Shop Scheduling Problem," ORSA Journal on Computing, v.3, n.2, pp.149-156, 1991.

Askin, R. G. and Standridge, C. R., "Modeling and Analysis of Manufacturing Systems," John Wiley & Sons, New York, 1993.

Atabakhsh, H., "A survey of constraint based scheduling systems using an artificial intelligence approach," Artificial Intelligence in Engineering, v. 6, n. 2, pp. 58-73, 1991.

Aytug, H., Koehler, G. J. and Snowdon, J. L., "Genetic Learning of Dynamic Scheduling within a Simulation Environment," Computers Ops. Res., vol. 21, n. 8, pp. 909-925, 1994.

Baillet, P. and Cauvin, A., "Proposal of a model of behavior for reactive scheduling systems," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.649-657, 1995.

Baker, K. R., "Introduction to Sequencing and Scheduling," John Wiley and Sons, Inc., New York, 1974.

Baker, K. R., "Sequencing rules and due-date assignments in a job shop," Management Science, v. 30, n.9, pp.1093-1104, 1984.

Barnes, J. W. and Chambers, J. B., "Solving the Job Shop Scheduling Problem with Tabu Search," IIE Transactions, vol. 27, pp. 257-263, 1995.

Barnes, J. W. and Laguna, M., "A tabu search experience in production scheduling," Annals of Operations Research, vol. 41, pp. 141-156, 1993.

Baxter, M. J., Tokhi, M. and Fleming, P. J., "Task-processor mapping for real-time parallel systems using genetic algorithms with hardwared-in-the-loop," IEE Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, pp.158-163, 1995.

Bean, J. C., "Genetic Algorithms and Random Keys for Sequencing and Optimization," ORSA Journal on Computing, vol. 6, n. 2 , pp. 154-160, 1994.

Beasley, J. E., "Lagrangian Relaxation," edited by Colin R. Reeves, John Wiley and Sons, Inc., New York, 1993.

Bector, C. R., Gupta, Y. P. and Gupta, M. C., "Determination of an optimal common due date and optimal sequence in a single machine job shop," Int. J. Prod. Res., v. 26, n. 4, pp. 613-628, 1988.

Bensana, E., Bel, G. and Dubois, D., "OPAL: A multi-knowledge-based system for industrial job-shop scheduling," Int. J. Prod. Res., v.26, n.5, pp.795-819, 1988.

Benton, W. C., "Time-based and cost-based priorities for job shop scheduling," Int. J. Prod. Res., v.31, n.7, pp. 1507-1519, 1993.

Bertrand, J. W. M., "The effect of workload dependent due-dates on job shop performance," Management Science, vol.29, n.7, pp. 799-816, 1983.

Bestwick, K. G. & Lockyer, P. F., "A pratical aprroach to production scheduling," Int. J. Prod. Res., vol.17, n.2, pp. 95-109, 1979

Biegel, J. E. and Davern, J. J., "Genetic Algorithm and Job Shop Scheduling," Computers Ind. Engng., vol. 19, n. 1-4, pp. 81-91, 1990.

Bierwirth, C., "A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms," site < http: // medusa.logistik.uni-bremen.de / Papers / Arbeitsberichte.html >, 1994

Bierwirth, C., Kopfer, H., Mattfeld, D. C. and Rixen, I., "Genetic Algorithm based Scheduling in a Dynamic Manufacturing Environment," ftp site < http:// medusa.logistik.uni-bremen.de / Papers / Arbeitsberichte.html >, 1995.

Bierwirth, C., Mattfeld D. C. and Kopfer H., "On Permutation Representations for Scheduling Problem," ftp site <http://medusa.logistik.uni-bremen.de/Papers/Arbeitsberichte.html>, 1996.

Blackstone, J. H., Phillips, D. and Hogg, G., "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," Int. J. Prod. Res., v.20, n.1, pp. 27-45, 1982.

Blazewicz, J., Ecker, K. H., Schmidt, G and Weglarz, J., "Scheduling in Computer and Manufacturing Systems," Springer-Verlag, 1994.

Blazewicz, J. and Finke, G., "Scheduling with Resource Management in Manufacturing Systems," European Journal of Operational Research, vol. 76, n.1, pp. 1-14, 1994.

Blume, C., "Planning and Optimization of Scheduling in Industrial Production by Genetic Algorithms and Evolutionary Strategy," Engineering Systems Design and Analysis, vol. 5, pp. 427-433, ASME, 1994.

Brennan, L. and Gupta, S. M., "A structured analysis of material requirements planning systems under combined demand and supply uncertainty," Int. J. Prod. Res., 1993.

Brucker, P., "Scheduling Algorithms," Springer, Berlin, 1995.

Bruns, R., "Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling," Proceedings of the 5th. International Conference on Genetic Algorithm, University of Illinois and Urbana-Chapaign,1993

Cândido, M., Bárcia, R, and Gauthier, F., "Uma revisão daprogramação da produção em ambientes job shop com fabricação por pedidos," Annals of XXVII SBPO, Vitória, Nov. 1995.

Cândido, M., "Local Search Algorithms and the Make-to-Order Job Shop Scheduling Problems," working paper, University of South Florida, 1996.

Carlier, J. and Pinson, E., "An algorithm for solving the job shop problem," Management Science, v.35, n.2, pp.164-176, 1989

Carlier, J. and Pinson, E., "Adjustment of heads and tails for the job-shop problem," European Journal of Operational Research, v.78, pp.146-161, 1994.

Cattrysse, D., Salomon, M., Kuik, R. and Wassenhove, L. N. V., "A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times," Management Science, v. 39, n. 4, April 1993.

Chang, C. Y. and Jeng, M. D., "Experimental Study of A Neural Model for Scheduling Job Shops," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, v.1, pp. 536, 1995.

Cheng, C. and Smith, S. F., "A constraint-posting framework for scheduling under complex constraits," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.269-280, 1995.

Chen, C. L. P., "Time Lower Bound for manufacturing Aggregate Scheduling Problems," Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento - CA, pp. 830-835, 1991.

Chen, H., Chiu, C. and Proth, J., "A More efficient Lagrangian Relaxation Approach to Job-Shop Scheduling Problems," IEEE International Conference on Robotics and Automation, pp.496-501, 1995.

Chen, T. and Hsia, T. S., "Job Shop Scheduling with Multiple Resources and an Application to A Semiconductor Testing Facility," Proceedings of 33rd. IEEE Conference on Decision and control, pp.1564-1570, 1994.

Chiu, C. and Yih, Y., "Learning-based methodology for scheduling in distributed manufacturing systems," International Journal of Production Research, v.33, n.11, pp. 3217-3232, 1995.

Crabtree, I. B., "Resource scheduling - comparing simulated annealing with constraint propagation," BT Technology J., v.13, n.1, pp.121127, 1995.

Croce, F. D., Tadei, R. and Volta, G., "A Genetic Algorithm for the Job Shop Problem," Computers Ops. Res., vol. 22, n. 1, pp. 15-24, 1995.

Czerwinski, C. S. and Luh, Peter B., "Scheduling Products with Bill of Materials Using an Improved Lagrangian Relaxation Technique," IEEE Transactions on Robotics and Automation, v.10, n.2, pp. 99-111, 1994.

Daniels, R. L. & Chambers, R. J., "Multiobjective fow-shop scheduling," Naval Research Logistics, vol.37, pp 981-995, 1990.

De, P., Ghosh, J. B. and Wells, C. E., "Optimal due-date assignment and sequencing," European Journal of Operational Research, v.57, pp. 323-331, 1992.

Dell'Amico, M. and Trubian, M., "Applying tabu search to the job-shop scheduling problem," Annals of Operations Research, v. 41, pp. 231-252, 1993.

Dobson, G. and Khosla, I "Simultaneous resource scheduling with batching to minimize weighted flow times," IIE Transactions (Institute of Industrial Engineers), v. 27 n. 5, pp. 587-598, 1995.

Doctor, S. R., Cavalier, T. M. and Egbelu, P. J., "Scheduling for machining and assembly in a job-shop environment," International Journal of Production Research vol. 31, n. 6, pp. 1275-1297,1993.

Dorndorf, U. and Pesch, E., "Evolution Based Learning in a Job Shop Scheduling Environment," Computers Ops. Res., vol. 22, n. 1, pp. 25-40, 1995.

Dondeti, V.R. & Emmons, H., "Algorithms for preemptive scheduling of different classes of processors to do jobs with fixed times," European Journal of Operational Research 70, 316-326, 1993.

Dorigo, M., Maniezzo, V. and Colorni, A., "Ant System: Optimization by a Colony of Cooperating Agents," IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, v.26, n.1, pp. 29-41, 1996.

Drummond, M., "Scheduling Benchmarks and Related Resources," FTP site <med@ptolemy.arc.nasa.gov>, 1996.

Dueck, G. and Scheuer, T., "Threshold Acceptance: a General Pupose Optimization Algorithm," Journal of Computational Physics, vol. 90, pp. 161-175, 1990.

Falkenauer, E. and Bouffouix, S., "A Genetic Algorithm for Job Shop," Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp. 824-829, Apr. 1991.

Fang, H. L., Ross, P., Corne, D., "A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems," Proceedings of the 5th. International Conference on Genetic Algorithm, University of Illinois and Urbana-Chapaign, 1993.

Fogel, D. B., "An Introduction to Simulated Evolutionary Optimization," IEEE Trans. on Neural Networks, vol. 5, n. 1, 1994.

Fox, M. S. and Smith, S. F., "ISIS - a knowledge-based system for factory scheduling," Expert Systems, v.1, n.1, pp. 25-49, 1984.

French, S., "Sequencing and Scheduling: An introduction to the Mathematics of the Job-Shop," Ellis Horwood Limited, England, 1982.

Gallone, J., Charpillet, F. and Alexandre, F., "Anytime Scheduling with Neural Networks," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.509-520, 1995.

Garey, M. R. and Johnson, D. S., "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman, San Francisco, 1979.

Gascon, A. & Leachman, R. C., "A dynamic programming solution to the dynamic, multi-item, single-machine scheduling problem," Operations Research, vol. 36, n.1, pp. 50-56, 1988.

Gauthier, F. A. O, "Programação da produção: uma abordagem utilizando algoritmos genéticos," doctoral thesis, UFSC - Brazil, 1993.

Ghosh, S., Melnyk, S. A. and Ragatz, G. L., "Tooling constraints and shop floor scheduling: evaluating the impact of sequence dependency," Int. J. Prod. Res., v.30, n.6, pp. 1237-1253, 1992.

Ghosh, S. and Gaimon, C., "Production Scheduling in a Flexible Manufacturing System with Setups," IIE Transactions, v.25, n.5, pp.21-35, 1993.

Giffler, B. and Thompson, G. L., "Algorithms for solving production-scheduling problems, " Operations Research, v.8, n.4, pp. 487-503, 1960.

Gilkinson, J. C., Rabelo, L. C. and Bush, B. O., "A Real-World Scheduling Problem Using Genetic Algorithms," Computers Ind. Engng., vol. 29, n. 1-4, pp. 177-181, 1995.

Glover, F et al, "Tabu Search: An introduction, technical aspects and applications in production and scheduling," Annals of Operations Research, v.41, pp. 1-279, 1993.

Glover, F. and Laguna M., "Tabu Search," Modern Heuristic Techniques for Combinatorial Problems, edited by Colin R. Reeves, John Wiley and Sons, Inc., New York, 1993.

Glover, F., Kelly, J. P. and Laguna, M., "Genetic Algorithms and Tabu Search: Hybrids for Optimization," Computers Ops Res., vol. 22, n. 1, pp. 111-134, 1995.

Goldberg, David E., "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley Publishing Company, 1989.

Goldratt, E. M., "Computerized shop floor scheduling," Int. J. Prod. Res., v. 26, n. 3, pp. 443-455, 1988.

Gonzalez, B., Torres, M. and Moreno, J. A., "A Hybrid Genetic Algorithm Approach for the No-Wait Flowshop Problem," IEE Conference Publication n.14, pp. 59-64, 1995.

Goyal, S.K., Sriskandarajah, C., "No-wait shop scheduling: computational complexity and approximate algorithms," OPSEARCH, vol.25, n.4, 220-244, 1988.

Graves, S. C., "A review of production scheduling," Operations Research, vol. 29, n.4, pp. 646-675, 1981.

Hamada, K., Baba, T., Sato, K. and Yufu, M., "Hybridizing a Genetic Algorithm with Rule-Based Reasoning for Production Planning," IEEE Expert, pp. 60-67, October 1995.

Hammer, P. L., "Annals of Operations Research: Production Planning and Scheduling," J.C. Baltzer AG Scientific Publishing Company, Bazel - Switzerland, 1990.

Han, W. and Dejax, P., "An Efficient Heuristic Based on Machine Workloads for Flowshop scheduling Problem with Setup and Removal," Laboratoire Economique, Industriel et Social, Ecole Centrale, Paris, 1-17, 1991.

Hastings, N. A. J., Marshall, P & Willis, R. J., "Schedule Based MRP: An integrated approach to production scheduling and material requirements planning," J. Op. Res. Soc., vol.33, pp 1021-29, 1982.

He, Z., Yang, T. & Deal, D. E., "A multiple-pass heuristic rule for job shop scheduling with due dates," Int. J. Prod. Res., 1993.

Herrmann, J. W., Lee, C. Y. and Hinchman, J., "Global Job Shop Scheduling with a GA," Production and Operations Management, vol. 4, n. 1, pp. 30-45, 1995.

Hertz, A. & Widmer, M. "A new heuristic method for the flow shop sequencing problem, European Journal of Operational Research, vol. 41, pp. 186-193, 1989.

Hoitomt, D. J. and Luh, P. B., "Scheduling the Dynamic Job Shop," Proceeding of IEEE Int. Conference on Robotics and Automation, v.3, pp. 71-76, 1993.

Hoitomt, D. J., Luh, P. B. and Pattipati, K. R., "A Practical Approach to Job-Shop Scheduling Problems," IEEE Transactions on Robotics and Automation, v.9, n.1, pp. pp.1-13, 1993.

Holsapple, C. W., Jacob, V. S., Pakath, R. and Zavery, S., "A Genetic-Based Hybrid Scheduler for Generating Static Schedules in Flexible Manufacturing Contexts," IEEE Trans. on Systems, Man, and Cybernetics, vol. 23, n. 4, pp. 953-972, 1993.

Homaifar, A., Qi, C. X. and Lai, S. H., "Constrainde Optimization Via Genetic Algorithms," Simulation, vol. 62, n. 4, pp. 242-254, 1994.

Hou, E. S. H. and Li, H. Y., "Task Scheduling for Flexible Manufacturing Systems based on Genetic Algorithms," IEEE , 397-402, 1991.

Hou, E. S. H., Ansari, N. and Ren, H., "A Genetic Algorithm for Multiprocessor Scheduling," IEEE Trans. on Parallel and Distributed Systems, vol. 5, n. 2, 1994.

Ikkai, Y., Inoue, M., Ohkawa, T. and Komoda, N., "A learning method of scheduling knowledge by genetic algorithm," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.641-648, 1995.

Itoh, K, Huang, D. & Enkawa, T., "Twofold look-ahead search for multi-criterion job shop scheduling," Int. J. Prod. Res., 1993.

Jansen, K., "Scheduling with constrained processor allocation for interval orders," Computers Ops. Res., v.20, n.6, pp. 587-595, 1993.

Jeffcoat, D. E. and Bulfin, R. L., "Simulated Annealing for Resource-Constrained Scheduling," European Journal of Operational Research, vol. 70, n. 1, pp. 43-51, 1993.

Johnson, S. M., "Optimal Two and Three Stage Production Schedules with Setup Times Included," Naval Research Logistics Quarterly, 1(1), 1954.

Kanet, J. J. and Adelsberger, H. H., "Expert Systems in Production Scheduling," European Journal of Operational Research, vol. 29, pp. 51-59, 1987.

Kannan, V. R. and Ghosh, S., "An evaluation of the interaction between dispatching rules and truncation procedures in job-shop scheduling," Int. J. Prod. Res., v.26, n.7, pp.1141-1159, 1993.

Kaplan, A. C. & Unal, A. T., "A probalistic cost-based due date assignment model for job shops," Int. J. Prod. Res.,1993.

Karp, R. M., "Combinatorics, complexity and randomness," Comm. of the ACM, v. 29, pp. 98-117, 1986.

Keer, R. M. and Ebsary, R. V., "Implementation of an expert system for production scheduling," European Journal of Operational Research, v.33, pp.17-29, 1988.

Khuri, S., Bäck, T. and Heitkötter, J., "An evolutionary approach to combinatorial optimization problems," ACM Computer Science Conference: proceedings, pp.66-73, 1994.

Kidwell, M. D. and Cook, D. J., "Genetic Algorithm for Dynamic Task Scheduling," IEEE 13th Annual Int. Phoenix Conf. On Computers and Communications, pp.61-67, 1994.

Kim, G. H. and Lee, C. S. G., "An evolutionary approach to the job-shop scheduling problem," Proceedings IEEE Int. Conference on Robotics and Automations, pp. 501-506, 1994.

Kim, G. H. and Lee, C.S.G., "Genetic Reinforcement Learning Approach to the Machine Scheduling Problem," Proceedings - IEEE International Conference on Robotics and Automation, v.1, pp. 196-201, 1995.

King, J.R. and Spachis, A.S.,"Scheduling: Bibliography and Review," Int. Jnl. of Physical Distribution and Material Management, vol. 10, n.3, pp.105-132, 1980.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P., "Optimization by Simulated Annealing," Science, vol. 220, n. 4598, pp. 671-680, 1983.

Kolen, A. and Pesch, E., "Genetic local search in combinatorial optimization," Discrete Applied Mathematics, v.48, pp.273-284, 1994.

Krishna, K., Ganeshan, K. and Ram, D. J., "Distributed Simulated Annealing Algorithms for Job Shop Scheduling," IEEE Trans.on Systems, Man, and Cybernetics, vol. 25, n. 7, 1995.

Laarhoven, P. J., Aarts, E. H. L. and Lenstra, J. K., "Job Shop Scheduling by Simulated Annealing," Operations Research, v. 40, n. 1, pp. 113-125, 1992.

Laborie, P. and Ghallab, M., "IxTeT: and Integrated Approach for Plan Generation and Scheduling," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.486-495, 1995.

Lee,I., Sikora, R., Shaw, M.J., "Joint lot sizing and sequencing with genetic algorithms for scheduling: evolving the chromosome structure," Proceedings of the 5th. International Conference on Genetic Algorithm, University of Illinois and Urbana-Chapaign, 1993.

Lee, K. and Jung, M., "Flexible Process Sequencing Using Petri Net Theory," Computers ind. Engng., v.28, n.2, pp. 279-290, 1995.

Lenstra, J. K. and Shmoys, D. B., "Computing Near-Optimal Schedules," Scheduling Theory and its Applications, Edited by P. Chrétienne, E. G. Coffman, J. K. Lenstra and Z. Liu, John Wiley & Sons Ltd, 1995.

Leon, V. J., Wu, S. D. and Storer, R. H., "Robustness measures and robust scheduling for job shops," IIE Transactions, v.26, n.5, pp.32-43, 1994.

Leu, Y., Matheson, L. A. and Rees, L. P., "Assembly Line Balancing Using Genetic Algorithms with Heuristic-Generated Initial Populations and Multiple Evaluation Criteria," Decision Sciences, v.25, n.4, 1994.

Liao, D., Chang, S., Yen, S. and Chien, C., "Daily Scheduling for R&D Semiconductor Fabrication," IEEE Int. Conf. On Robotics and Automation proceedings, v.3, pp. 77-82, 1993.

Lin, C. K. Y., Haley, K. B. and Sparks, C., "A comparative study of standard and adaptive versions of threshold acceptance and simulated annealing algorithms in three scheduling problems," European Journal of Operational Research, vol. 83, pp. 330-346, 1995.

Lin, F.T., Kao, C.Y, Hsu, C.C., "Applying the genetic approach to simulated annealing in solving some NP-Hard Problems," IEEE transactions on systems, man, and cybernetics, vol. 23, n. 6, 1752-1767, 1993.

Lo, Z. and Bavarian, B, "Scheduling with Neural Networks for Flexible Manufacturing Systems," Proceedings of the 1991 IEEE International Conference on Robotics and Automation," pp.818-823, 1991.

Lourenco, H. R., " Job shop scheduling: computational study of local search and large-step optimization methods," European Journal of Operational Research, v. 83, pp. 347 - 364, 1995.

Low, C., "Job Shop Scheduling Heuristics for Sequence Dependent Setups," Computers Ind. Engng., vol. 29, n. 1-4, pp. 279-283, 1995.

Luh, P. B. and Hoitomt, D. J., "Scheduling of Manufacturing Systems Using the lagrangian Relaxation Technique," IEEE Transactions on Automation Control, v.38, n.7, pp. 1066-1079, 1993.

Maccarthy, B.L. & Liu, J., "Adrressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling," Int. J. Prod. Res.,1993

Maqrini, H. El and Teghem, J., "Scheduling complex flexible job shop problems," Proceedings of IEEE International Workshop on Emerging Technologies and Factory Automation, Paris, pp. 541-549, 1995

Matsuura, H., Tsubone, H. and Kanezashi, M., "Sequencing, dispatching, and switching in a dynamic manufacturing environment," Int. J. Prod. Res., 1993.

Mattfeld, D. C., "Evolutionary search and the Job Shop," Physica-Verlag Heidelberg, Germany, 1996.

Mattfeld, D. C. and Bierwirth, C., "A Search Space Analisys of the Job Shop Scheduling Problem," site < http: // medusa.logistik.uni-bremen.de / Papers / Arbeitsberichte.html >, 1996.

McAreavey, D., Hoey, J. and Leonard, R., "Designing the closed loop elements of a material requirements planning system in a low volume, make-to-order company (with case study)," Int. J. Prod. Res., v.26, n.7, pp.1141-1159, 1988.

McCahon, C.S and Lee, E. S., "Fuzzy job sequencinc for a flow shop". European Journal of Operational Research , v.62, pp.294-301, 1992.

Mckay, K. N., Safayeni, F. R., Buzacon, J. A., "Job-shop scheduling theory: What is releant?," Interfaces , 18: 4, pp 84-90, 1988.

Meyer, J., "Towards effective decision support for production scheduling," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.523-532, 1995.

Moreno, A. A. & Ding, F. Y., "A constructive heuristic algorithm for concurrently selecting and sequencing jobs in an FMS environment," Int. J. Prod. Res., 1993.

Müller-Merbach, H., "Heuristics and their design: a servey," European Journal of Operational Research v.8, pp.1-23, 1981.

Musser, K. L., Dhingra, J. S. and Blankenship, G. L., "Optimization Based Job Shop Scheduling," IEEE Trans. on Automatic Control, vol. 38, n. 5, pp. 808-813, 1993.

Nakamura, N. and Salvendy, G., "An experimental study of human decision-making in computer-based scheduling of flexible manufacturing system," Int. J. Prod. Res., vol.26, n.4, pp. 567-583, 1988.

Nakano, R., Davidor, Y. and Yamada, T., "Conventional Genetic Algorithm for Job Shop Problems," edited by Y. Davidor, 1994.

Nasr, N., "Scheduling of production systems with flexible routings," IIE 2nd. Industrial Engineering Research Conference Proceedings, Los Angeles - CA, pp. 487-492, 1993.

Nordström, A-L. and Tufekci, S., "A Genetic Algorithm for the Talent Scheduling Problem," Computers Ops. Res., v.21, n.8, pp. 927-940, 1994.

Özdamar, L. and Ulusoy, G., "A survey on the resource-constrained project scheduling problem," IIE Transactions, v. 27, pp. 574-586, 1995.

Park, L. J. and Park, C. H., "Genetic Algorithm for Job Shop Scheduling Problems Based on Two Representational Schemes," Eletronics Letters, vol. 31, n. 9, pp. 2051-2053, 1995.

Pape, C. L., "Three mechanisms for managing resource constraints in a library for constraint-based scheduling," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.281-289, 1995.

Park, L. J. and Park, C. H., "Application of Genetic Algorithm to Job Shop Scheduling Problems with Active Schedule Constructive Crossover," Proceedings of IEEE International Conference on Systems, Man and Cybernetics, v.1, pp.530-535, 1995.

Parker, R. G., "Deterministic Scheduling Theory," Chapman & Hall, London, 1995.

Pinson, E., "The Job Shop Scheduling Problem: A Concise Survey and Some Recent Developments," Scheduling Theory and its Applications Edited by P. Chretienne, E. G. Coffman, J. K. Lenstra and Z. Liu, John Wiley and Sons Ltd, 1995.

Piramuthu, S., Raman, N. and Shaw M., "Learning-Based Scheduling in a Flexible Manufacturing Flow Line," IEEE Transactions on Engineering Management, v. 41, n. 2, pp. 172-182, 1994.

Poon, P. W. and Carter, J. N., "Genetic algorithm crossover operators for ordering applications," Computers Ops. Res., v. 22, n. 1, pp. 135-147, 1995.

Punnen, A. P. and Aneja, Y. P., "Categorized Assignment Scheduling: a Tabu Search Approach," J. Opl. Res. Soc., v.44, n.7, pp. 673-679, 1993.

Quaddus, M. A., "A Generalized Model of Optimal Due-Date Assignment by Linear Programming," J. Opl. Soc., vol 38, n. 4, pp 353-359, 1987.

Quinlan, J. R., "Decision trees and decision making," IEEE Trans. Syst., Man, Cybernetics, vol. 20, pp. 339-346, 1990.

Rabelo, L., Yih, Y., Jones, A. and Tsai, J., "Intelligent Scheduling forFlexible Manufacturing Systems," Proceedings of IEEE Int. Conf. on Robotics and Automation, v.3, pp. 810-815, 1993.

Rachamadugu, R., "Scheduling jobs with proportionate early/tardy penalties," IIE Transactions, vol.27, pp. 679-682, 1995.

Raman, N. and Talbot, F. B., "The job shop tardiness problem: a decomposition approach," European Journal of Operational Research, vol. 69, pp. 187-199, 1993.

Reeves, C. R., "Modern Heuristic Techniques for Combinatorial Problems," John Wiley & Sons, New York, 1993.

Reeves, C. R., "A Genetic Algorithm for Flowshop Sequencing," Computers Ops. Res., v.22, n.1, pp.5-13, 1995.

Richard, P, Jacquet, N., Cavalier, C. and Proust, C., "Solving Scheduling problems Using Petri Nets and Constraint Logic Programming," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.59-67, 1995.

Rodammer, F.A. & White, K.P., "A recent survey of production scheduling," IEEE transactions on systems, man, and cybernetics, vol. 18, n. 6, 841-851, 1988.

Rodolph, G., "Massively Parallel Simulated Annealing and Its Relation to Evolutionary Algorithms," The Massachusetts Institute of Tecnology - Evolutionary Computation vol.1, n.4, pp.361-383, 1994.

Rubin, P. A. and Ragatz, G. L., "Scheduling in a Sequence Dependent Setup Environment with Genetic Search," Computers Ops. Res., vol. 22, n. 1, pp. 85-99, 1995.

Rudolph, G., "Massively Parallel Simulated Annealing and Its Relation to Evolutionary Algorithms," Evolutionary Computation, 1(4), pp. 361-383, 1994.

Sarin, S. C., Ahn, S. and Bishop, A. B., "An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime," Int. J. Prod. Res., vol. 26, n.7, pp. 1183-1191, 1988.

Satake, T., Morikawa, K. and Nakamura, N., "Neural network approach for minimizing the makespan of the general job-shop," International Journal of Production Economics, v.33, pp.67-74, 1994.

Schrape, L. & Baker, K. R., "Dynamic Programming Solution of Sequencing Problems with Precedence Constraints," Operational Research, vol. 26, n. 3, pp.444-450, 1978.

Sen, T. and Gupta, S. K., "Astate-of-art survey of static scheduling research involving due dates," The Int. Jl. of Mgmt. Sci., vol. 12, n.1, pp 63-76, 1984.

Serafini, P. and Speranza, M. G., "A decomposition approach for a resource constrained scheduling problem," European Journal of Operational Research, v.75, pp. 112-135, 1994.

Shen, C., Pao, Y. and Yip, P., "Scheduling Multiple Job problems with Guided Evolutionary Simulated Annealing Approach," Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 702-706, 1994.

Shmoys, D. B., Clifford, S. and Wein. J., "Improved Approximation Algorithms for Shop Scheduling Problems," SIAM J. COMPUT., v.23, n.3, pp. 617-632, 1994.

Schrage, L. and Baker, K. R., "Dynamic Programming Solution of Sequencing Problems with Precedence Constraints," Operations Research, v.26, n.3, pp. 444-449, 1978.

Shimoyashiro, S., Isoda, K. and Awane, H., "Input scheduling and load balance control for a job shop," Int. J. Prod. Res., v.22, n.4, pp.597-605, 1984.

Sim, S. K., Yeo, K. T. and Lee, W. H., "An expert neural network system for dynamic job shop scheduling," Int. J. Prod. Res., 1994.

Smith, S. F., "OPIS: A Methodology and Architecture for Reactive Scheduling," Intelligent Scheduling, edited by M. Zweben and M. S. Fox, Morgan Kaufmann Publishers, San Francisco - CA, 1994.

Sponsler, J.L., "Genetic algorithms applied to the scheduling of the hubble space telescope," Telematics, vol.6, n.3/4, pp. 181-190, 1989.

Stöppler, S. and Bierwirth, C., "The Application of a Parallel Genetic Algorithm to the n/m/P/Cmax Flowshop Problem," < http: // medusa.logistik.uni-bremen.de / Papers / Arbeitsberichte.html >, 1995.

Storer, R. H., Wu, S. D. and Vaccari, R., "New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling," Management Science, vol. 38, n. 10, pp. 1495-1509, 1992.

Sule, D. R., "Industrial Scheduling," PWS Publishing Company, Boston, 1997.

Sun, D., Batta, R. and Lin, L., "Effective job shop scheduling through active chain manipulation," Computers Ops. Res., v.22, n.2, pp.159-172, 1995.

Syswerda, G., "Schedule Optimization Using Genetic Algorithms," Handbook of Genetic Algoritthms, Van Nostrand Reinhold, New York, pp. 332-349, 1991.

Taillard, E., "Benchmarks for basic scheduling problems," European Journal of Operational Research, v.64, pp. 278-285, 1993.

Tang, C. S., "The impact of uncertainty on a production line," Management Science, v.36, n.12, pp. 1518-1531, 1990.

Tsang, E. P. K., "Scheduling techniques - a comparative study," BT Technol. J., v.13, n.1, pp.16-28, 1995.

Uckum, S., Bagachi, S. and Kawamura, K., "Managing genetic search in job shop scheduling," IEEE Expert, vol.8, n.5, pp.15-24, 1993.

Vaca, O. C. L., "Um algoritmo evolutivo para a programação de projetos multi-modos com nivelamento de recursos limitados," doctoral thesis, UFSC, 1995

Venugopal, V. and Narendran, T. T., "A genetic Algorithm Approach to the Machine-Component Grouping Problem with Multiple Objectives," Computers ind. Engng., v.22, n.4, pp.469-480, 1992.

Wang, D., "Earliness/tardiness production planning approaches for manufacturing systems," Computers ind. Engng., v.28, n.3, pp. 425-436, 1995.

Werner, F., "On heuristic solution of the permutation flow shop problem by path algorithms," Computers Operations Research, vol.10/7, pp.707-722, 1993.

White, K. P. and Rogers, R. V., "Job-shop scheduling: limits of the binary disjunctive formulation," Int. J. Prod. Res., v.28, n.12, pp. 2187-2200, 1990.

Whitley, D., "The traveling salesman and sequence schedule scheduling: quality solutions using genetic edge recombination," Handbook of Genetic Algoritthms, Van Nostrand Reinhold, New York, pp. 350-372, 1991.

Widmer, M. and Hertz, A., "A new heuristic method for the flow shop sequencing problem," European Journal of Operational Research, v.41, pp. 186-193, 1989.

Widmer, M., "Job Shop Scheduling with Tooling Constraints: a Tabu Search Approach," J. Opl. Res. Soc., vol. 42, n. 1, pp 75-82, 1991.

Wu, H.-H. and Li, R.-K., "A new rescheduling method for computer based scheduling systems," Int. J. Prod. Res., v.33, n.8, pp. 2097-2110, 1995.

Wu, S. D., Storer, R. H. & Chang, P. C., "One-machine rescheduling heuristics with efficiency and stability as criteria," Computers Ops. Res., vol.20, n. 1, pp 1-14, 1993

Xiong, H. H., "Scheduling flexible manufacturing systems based on timed Petri nets and fuzzy dispatching rules," IEEE Symposium on Emerging Technologies and Factory Automation, v.1, pp.309-315, 1995.

Yamada, T. and Nakano, R., "Genetic algorithm with multi-step crossover for job shop scheduling problems," IEE Conference Publications n. 414, pp. 146-151, 1995.

Zweben, M. and Fox, M. S., "Intelligent Scheduling," Morgan Kaufmann Publishers, San Francisco - CA, 1994.

APPENDIX A
RANDOM PROBLEM GENERATOR PROGRAM

A random problem generator program was developed in order to obtain problem instances that included the real world constraints considered by the system proposed in this thesis. Given the probability distributions for some variables and values for some parameters the program generates a job shop scheduling problem like the ones described in appendix A. There is also an option to enter all the data from a specific problem, but this is very time consuming. The random generator program was used to create all the problem used in this thesis.

When the program requests the probability distribution of a variable the user must choose among binomial, Poisson, normal, uniform or exponential distributions and enter the distribution parameters (e.g., mean and variance for a normal distribution, minimum and maximum for a uniform distribution, etc.). Using this distribution the program randomly generates values to the variable. The inputs that must be provided by the user to the program generator are described below:

1. Number of jobs
2. Probability distribution of "Number of parts per job"
3. Occurrence probability of job ready times
4. Probability distribution of "Job batch size"
5. Probability distribution of "Ratio of a part batch size over the batch size of its succeeding part in the BOM"
6. Probability distribution of "Job ready time (given that it occurs)"
7. Probability distribution of "Number of parts preceding a part in a BOM"
8. Probability distribution of "Number of subprocess per part"
9. Probability distribution of "Number of routes per subprocess"
10. Probability distribution of "Number of operations per route"
11. Probability of occurrence of operation ready time
12. Probability distribution of "Operation ready time (given that it occurs)"
13. Probability distribution of "Ratio of part batch size over transport batch size"

14. Probability distribution of "Unit operation execution time"
15. Probability distribution of "Operation setup time"
16. Number of machines which execute sequence dependent setup operations
17. Number of machine cells performing no sequence dependent setup operations
18. Probability distribution of "Number of machines per machine cell"
19. Number of resource types other than machines
20. Probability distribution of "Time interval between two consecutive resource scheduled breaks"
21. Probability distribution of "Resource break time duration"
22. Mean machine static loading rate, based on historical data
23. Increase in the machine load due to the absence of resources other than machines
24. Mean number of resource types (other than machine) used to execute an operation
25. Probability distribution of "Number of units of each resource type used to perform an operation (given that the resource type is required)"
26. Probability distribution of "Probability of the number of units of a resource type required at a time t be greater than the number of resources available"
27. Coefficient of variation of the percentage of operations that require a specific resource type (other than machine)
28. Probability of an operation presents setup dependent of at least one other operation, given that it is executed on sequence dependent setup machine
29. Given that an operation 'o' has sequence dependent setup, enter the probability of other operation which is executed on the same machine modify the standard setup time of 'o' if processed right before 'o'
30. Probability distribution of "Job due dates"
31. Overlap policy (1/2)

Inputs number 1 (number of jobs), 2 (probability distribution of "Number of parts per job") and 7 (Probability distribution of "Number of parts preceding a part in a BOM") are used to create the BOMs of all jobs. A tree generation algorithm was implemented to accomplish this task.

The probability distributions of "Job batch size" (item 4) and "Ratio of a part batch size over the batch size of its succeeding part in the BOM" (item 5) are utilized to determine the batch size for each part.

The probability distributions of "Number of subprocess per part", "Number of routes per subprocess" and "Number of operations per route" are useful to define all the alternative processing plans of each part, i.e., to define the sequence of subprocesses for each part, the alternative routes per subprocess, and the sequence of operations at each subprocess route.

Items 16, 17 and 18 determine the machinery in the manufacturing environment. Notice that if a machine executes sequence dependent setup operations it is modeled as an one-machine cell. Therefore identical machines are included in the same cell if they execute no sequence dependent setup operation, and in separated ceils otherwise.

The number of machines that execute at least one sequence dependent setup operation (item 16), the probability of an operation (executed on a sequence dependent setup machine) to present setup dependent of at least one other operation (item 28), and the probability of an operation (which is executed on the same machine of an operation 'o' with sequence dependent setups) modify the standard setup time of 'o' if processed right before 'o' (item 29) are used to define the set of operations with sequence dependent setups. All these inputs are required because not all operations processed on a sequence dependent setup machine present sequence dependent setup times.

The number of units of each resource type (other than machine) in the production environment, as well as the resource requirements for each operation are jointly determined by inputs 22 through 27. By using these inputs the user can create problems with different levels of competition for resources, and also different levels of availability and utilization of each resource type.

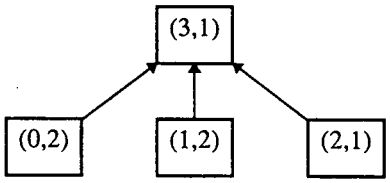
The uses of the other inputs are quite obvious.

APPENDIX B
PROBLEM INSTANCES USED IN THE EXPERIMENTS

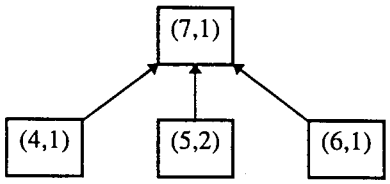
The six problem instances used in the experiments addressed in chapter 5 are described here. They were generated by the random generator program described in appendix A. The problems are proposed as benchmarks for real job-shop scheduling.

Problem 1: The bill of material (BOM) of each job is shown in figure A.1. A pair (i, j) inside each box means (part i, number of units of part i required to produce one unit of the final product).

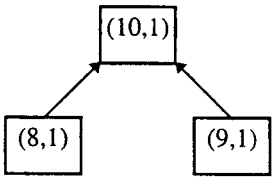
Job 0:



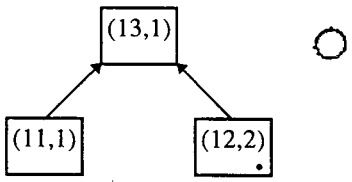
Job 1:



Job 2:



Job 3:



Job 4:

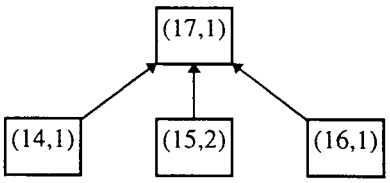


Figure A.1. BOMs - Jobs from Problem 1

Table A.1 shows some job related data. Table A.2 describes the environment capacity (availability of resources) and the resource scheduled breaks. Routing information and operation related data, including resource requirements are described in table A.3. The subset of operations with sequence dependent setup is reported in table A.4.

Job	batch size	due date	ready time
0	6	515	0
1	11	456	0
2	12	443	0
3	9	475	0
4	14	574	0

Table A.1. Job Related Dada - Problem 1

Resource class	Resource (type|unit)	Scheduled breaks (start time|duration)
Machines	(0|0)	(335|56) (1781|22)
	(1|0)	(506|54) (1630|36)
	(2|0)	(1053|1)
	(3|0)	(905|1)
	(3|1)	(612|14) (1986|16)
	(4|0)	(528|1) (1499|18)
	(5|0)	(291|36) (1717|6)
	(5|1)	(936|2)
	(5|2)	(223|30) (1663|36)
	(5|3)	(1181|78)
Other resources	(6|0)	(673|30)
	(6|1)	(1269|14)
	(6|2)	(113|36) (1539|66)
	(6|3)	(311|34) (1545|48)
	(7|0)	(480|2) (1602|1)
	(7|1)	(426|50) (1616|26)
	(7|2)	(767|16)
	(7|3)	(1238|46)
	(8|0)	(955|32)
	(8|1)	(403|34) (1597|14)
	(9|0)	(281|58) (1359|32)
	(9|1)	(842|40)
	(10|0)	(714|30)
	(10|1)	(1156|48)
	(10|2)	(866|40)
	(10|3)	(771|36)
	(11|0)	(124|1) (1185|50)
	(11|1)	(905|54)
	(11|2)	(127|2) (1379|28)
	(11|3)	(1041|20)
	(12|0)	(244|16) (1380|36)

	(12|1)	(378|24) (1682|26)
	(12|2)	(59|48) (1237|48)
	(12|3)	(149|1) (1400|50)
	(13|0)	(572|1) (1713|24)
	(13|1)	(510|26) (1686|18)
	(13|2)	(964|14)
	(13|3)	(181|6) (1557|46)

Table A.2. Resource Related Data - Problem 1

Part	Sub-process	Route	Operation	Minimum transport batch	Setup time	Unit execution time	Operation ready time	Machine type required	Other resources required (typelquantity)
0	0	0	0	4	6	2	0	2	(8|1)
0	0	0	1	5	6	3	0	3	(6|1)
0	0	0	2	3	7	1	0	2	(6|1) (7|1) (13|1)
0	1	1	3	6	3	2	0	3	(8|2)
0	1	1	4	3	3	4	0	5	
1	2	2	5	4	3	1	0	5	(10|1)
1	2	2	6	3	10	3	0	0	
1	3	3	7	3	4	3	0	1	(9|2)
1	3	4	8	10	6	1	0	4	
1	3	4	9	9	4	1	0	5	
1	3	4	10	3	6	2	0	2	(10|1) (12|2)
2	4	5	11	4	6	3	0	4	(13|1)
2	4	5	12	5	6	3	0	5	(7|2) (10|1) (13|1)
2	4	6	13	5	3	1	0	5	(7|1) (10|1)
2	4	6	14	2	5	2	0	1	
3	5	7	15	2	4	2	0	0	(8|1)
3	5	7	16	5	8	1	0	3	(7|1) (8|1) (10|1) (11|1)
3	5	8	17	2	5	5	0	0	(8|1) (10|1)
3	5	8	18	3	6	1	0	4	
3	6	9	19	2	5	4	0	5	(7|1) (10|1)
3	6	10	20	3	6	3	0	2	(7|1)
3	6	10	21	2	6	4	0	3	(6|2) (11|1) (12|1)
4	7	11	22	2	7	4	0	3	
4	7	11	23	8	5	2	0	5	
4	7	12	24	3	4	4	0	1	(6|1) (7|1) (11|1)
4	7	12	25	4	3	5	0	4	(7|1)
4	8	13	26	4	9	4	98	2	(7|1) (10|1)
4	8	14	27	8	6	1	0	3	(10|1) (13|1)

4	9	15	28	2	4	1	0	4	(13|1)
4	9	16	29	7	6	4	0	1	
4	9	16	30	6	4	2	0	3	(6|1)
4	10	17	31	5	5	5	0	2	(8|1) (10|1)
4	10	17	32	4	4	1	0	3	
4	10	17	33	4	4	5	0	0	(10|1)
5	11	18	34	11	2	2	0	4	(6|2) (9|1)
5	12	19	35	5	8	1	0	3	(11|1) (12|1)
5	12	19	36	8	3	4	0	5	(6|2) (7|1)
5	12	20	37	5	4	5	0	5	(6|1)
5	13	21	38	10	2	5	0	2	(11|1)
5	13	21	39	7	3	4	0	4	
6	14	22	40	4	8	5	0	5	(6|1) (10|1)
6	14	22	41	3	4	2	0	0	
6	14	22	42	5	4	1	0	3	(7|1) (11|1)
6	15	23	43	2	3	3	0	2	
6	15	23	44	3	8	3	0	4	
6	15	23	45	5	6	3	0	2	
6	15	24	46	2	1	3	0	5	
6	15	24	47	9	8	5	0	4	(7|1) (11|1)
6	15	24	48	2	5	1	0	1	
6	16	25	49	3	2	1	0	4	(12|2) (13|1)
7	17	26	50	7	2	4	0	1	(6|1)
7	18	27	51	10	7	4	0	4	(13|1)
7	18	28	52	6	8	6	0	4	
7	18	28	53	10	5	1	0	3	(7|1) (10|1)
7	19	29	54	9	1	5	0	2	(8|1) (13|1)
7	19	29	55	5	5	3	0	5	(7|1) (10|1)
7	19	29	56	3	3	5	0	2	
8	20	30	57	3	8	2	0	5	(7|1) (10|1)
8	20	30	58	8	4	3	0	2	(6|1)
8	20	31	59	6	6	1	0	2	(7|1) (10|1)

8	20	31	60	3	6	3	0	5	(7|1) (12|1)
8	21	32	61	5	6	5	0	3	(10|1)
8	21	32	62	3	7	5	0	4	
8	21	33	63	4	5	2	0	4	(7|1)
9	22	34	64	8	8	2	0	2	(6|1) (8|2)
9	22	35	65	2	5	4	0	3	(11|1)
9	22	35	66	2	6	5	0	2	(7|1)
9	22	35	67	3	6	3	0	3	(11|1) (12|1)
9	23	36	68	3	5	4	0	0	(13|1)
9	23	36	69	3	3	5	0	4	
9	24	37	70	3	6	1	0	2	(10|1) (13|2)
10	25	38	71	4	6	2	0	5	(6|1)
10	25	38	72	3	7	2	0	2	(9|1) (11|1)
10	26	39	73	3	5	3	0	3	(7|1)
10	26	39	74	6	7	2	0	5	
10	27	40	75	6	7	3	0	3	(10|1) (13|1)
10	27	40	76	3	7	2	0	0	(12|1)
10	27	41	77	3	3	4	0	3	(7|1) (10|1)
10	27	41	78	4	5	1	0	2	(7|1)
11	28	42	79	2	8	2	0	4	(13|1)
11	28	42	80	2	6	5	0	0	
11	28	42	81	3	1	2	0	3	(10|1)
11	28	43	82	5	5	2	0	4	(10|1)
11	28	43	83	3	8	3	0	0	(6|1)
11	29	44	84	3	6	1	0	1	(13|1)
12	30	45	85	7	1	3	0	5	(6|1)
12	31	46	86	4	6	2	0	3	
12	31	46	87	12	5	1	0	2	(6|1)
12	31	47	88	5	3	1	0	2	(6|1) (12|1)
12	32	48	89	8	8	5	0	3	(6|2) (7|1)
12	32	48	90	7	9	2	0	5	
12	32	49	91	4	4	1	0	1	(6|1)

12	32	49	92	10	7	2	0	4	(10|1) (12|1)
13	33	50	93	5	2	1	0	3	
13	33	50	94	5	10	2	0	2	(10|1)
13	33	50	95	3	4	6	0	5	(10|1) (12|1)
13	34	51	96	3	9	4	0	4	(8|2)
13	34	51	97	5	6	4	0	3	(6|1)
13	35	52	98	5	5	2	0	1	(7|1) (8|1) (9|2) (10|1) (11|1)
13	35	52	99	4	4	1	0	4	(13|1)
13	36	53	100	5	7	4	0	1	
13	36	53	101	2	3	4	0	5	(7|2)
14	37	54	102	4	3	3	0	5	(6|2) (7|1) (8|1) (10|1) (11|1)
14	38	55	103	5	2	6	0	3	
14	38	55	104	14	5	5	0	5	
14	38	56	105	4	8	2	0	3	(11|1)
14	38	56	106	10	3	4	0	5	
14	39	57	107	7	6	2	0	2	(8|1) (11|1)
14	40	58	108	6	3	3	0	5	(7|1) (11|1) (13|1)
14	40	58	109	3	3	5	0	1	
14	41	59	110	6	5	1	0	0	
14	41	59	111	9	5	1	0	4	
14	41	59	112	5	6	2	0	1	(11|1)
14	41	60	113	6	3	5	0	0	
14	41	60	114	11	6	1	0	2	(7|1)
14	41	60	115	7	3	1	0	5	
15	42	61	116	13	4	3	0	1	(11|1)
15	42	61	117	23	4	1	0	5	(11|1)
15	42	62	118	9	6	2	0	4	
15	43	63	119	9	8	2	0	3	(6|1)
15	43	63	120	26	5	1	0	1	(6|2) (13|1)
15	43	64	121	10	5	1	0	3	(11|1)

15	44	65	122	6	1	1	0	5	(6|1) (7|1) (13|1)
16	45	66	123	3	3	3	0	2	(12|2)
16	46	67	124	3	6	2	0	0	
16	46	67	125	7	4	1	0	3	(10|2)
16	46	67	126	12	7	1	0	1	(11|1)
17	47	68	127	5	9	1	0	2	(8|1) (10|1) (11|1)
17	47	68	128	4	4	6	0	5	(7|1) (8|1) (13|1)
17	48	69	129	4	7	3	0	2	
17	48	69	130	3	5	5	0	5	
17	48	69	131	5	3	2	0	2	(10|1) (11|1) (12|1)
17	49	70	132	5	5	2	0	4	
17	49	70	133	3	7	4	0	2	(6|1) (10|1)
17	50	71	134	5	5	3	0	3	(11|1)
17	50	71	135	7	3	1	0	4	(7|2) (10|1) (12|1)

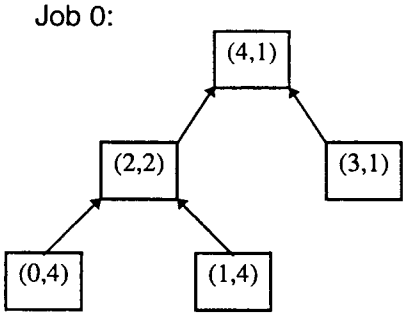
Table A.3. Routing Structure and Operation Related Data - Problem 1

Operation	Setup times (preceding operation | setup)
6	(15|7) (17|5) (33|5) (41|5) (68|6) (76|8) (80|6) (110|5) (124|9)
14	(24|4) (48|1) (50|3) (91|3) (98|6) (100|6) (109|4) (112|4) (116|4)
15	(6|9) (17|3) (68|5) (76|7) (110|5) (124|3)
17	(6|7) (15|10) (33|4) (41|4) (68|3) (76|6) (80|6) (124|3)
24	(14|7) (29|5) (48|7) (50|7) (91|0) (98|3) (112|7) (116|7)
29	(14|4) (48|5) (50|7) (98|5) (100|4) (109|5) (112|5) (116|9) (120|3)
33	(6|6) (17|5) (41|5) (68|6) (76|6) (80|3) (110|7) (124|2)
41	(6|7) (17|2) (33|6) (68|8) (76|5) (80|2) (110|7) (124|6)
48	(14|2) (24|4) (50|4) (91|2) (98|6) (109|4) (112|8) (116|4) (120|1)
50	(24|4) (29|7) (48|4) (98|5) (100|6) (109|5) (116|5) (120|4)
68	(6|6) (15|6) (17|0) (33|7) (41|3) (76|3) (80|3) (110|7) (124|8)
76	(6|4) (15|3) (17|3) (33|4) (41|3) (68|5) (80|5) (110|2) (124|3)
80	(6|5) (15|3) (17|5) (41|8) (68|7) (110|3)
91	(14|7) (24|2) (29|7) (50|3) (98|6) (116|6) (120|6)
98	(14|4) (24|4) (29|4) (48|6) (50|2) (91|3) (100|3) (109|8) (112|7) (116|7)
100	(14|6) (24|2) (29|6) (48|3) (50|4) (91|5) (109|5) (116|4) (120|5)
109	(14|5) (24|5) (48|6) (50|7) (91|5) (98|0) (100|10) (112|5) (116|2) (120|5)
110	(6|8) (17|3) (33|6) (41|4) (68|6) (76|3) (80|4) (124|8)
112	(14|4) (24|2) (29|4) (48|8) (50|3) (109|3) (116|8) (120|2)
116	(24|6) (29|3) (48|10) (50|5) (98|2) (100|3) (109|7) (112|3) (120|2)
120	(14|6) (24|7) (29|2) (48|8) (50|7) (91|9) (98|6) (100|4) (109|6) (112|4) (116|7)
124	(17|3) (41|5) (68|4) (80|4) (110|9)

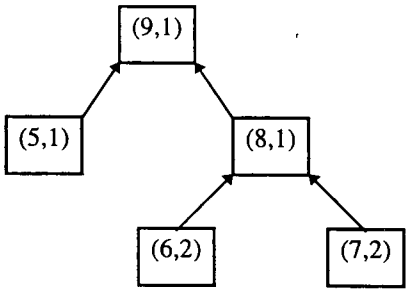
Table A.4. Operations with Sequence Dependent Setup Times - Problem 1

Problem 2: Similarly figure A.2 and tables A.5, A.6, A.7 and A.8 describe problem 2.

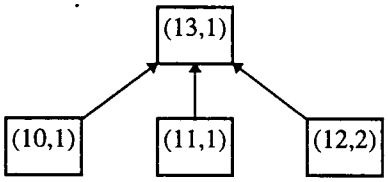
Job 0:



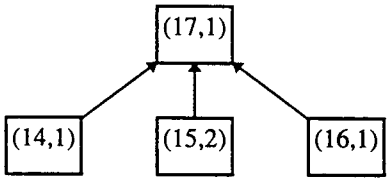
Job 1:



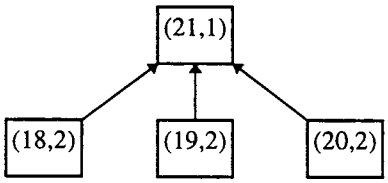
Job 2:



Job 3:

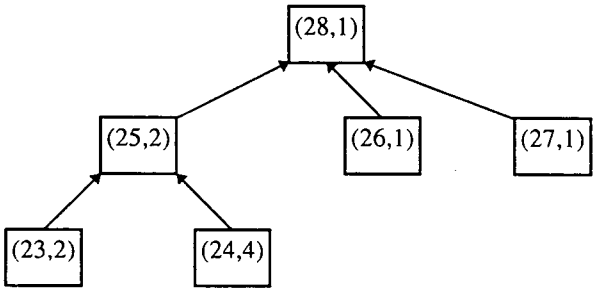


Job 4:



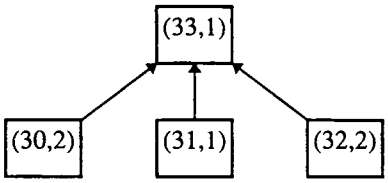
Job 5:

Job 6:



Job 7:

Job 8:



Job 9:

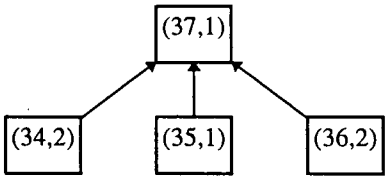


Figure A.2. BOMs - Jobs from Problem 2

Job	batch size	due date	ready time
0	13	1120	0
1	11	1768	0
2	10	2632	0
3	8	940	0
4	10	1804	0
5	13	2812	0
6	9	1156	0
7	11	1732	0
8	9	1876	0
9	10	2956	0

Table A.5. Job Related Dada - Problem 2

Resource class	Resource (type|unit)	Scheduled breaks (start time|duration)
Machines	(0|0)	(1667|27)
	(1|0)	(1235|30) (4235|31)
	(2|0)	(2499|17)
	(3|0)	(166|25) (3071|27)
	(3|1)	(1823|32)
	(4|0)	(832|37) (3779|29)
	(4|1)	(2981|34)
	(5|0)	(1032|14) (4316|30)
	(6|0)	(2074|27)
	(6|1)	(1310|42) (4562|20)
	(6|2)	(2590|40)
	(7|0)	(1680|41)
	(7|1)	(2881|10)
	(7|2)	(56|31) (3147|29)
	(8|0)	(2278|19)
	(8|1)	(887|43) (4080|13)
	(8|2)	(2970|38)
	(9|0)	(2331|43)
	(9|1)	(1654|24)
	(9|2)	(1174|32) (3996|22)
	(10|0)	(2399|45)
	(10|1)	(1082|24) (3806|39)
Other resources	(11|0)	(179|30) (3629|20)
	(11|1)	(2079|27)
	(11|2)	(297|40) (2827|25)
	(11|3)	(2266|23)
	(12|0)	(928|27) (3775|26)
	(12|1)	(2576|28)
	(12|2)	(932|39) (3371|37)
	(12|3)	(1363|45) (4828|38)
	(13|0)	(589|22) (3971|1)

	(13|1)	(2764|23)
	(13|2)	(106|18) (3124|18)
	(13|3)	(27|42) (3639|29)
	(14|0)	(27|32) (3209|35)
	(14|1)	(1779|31)
	(15|0)	(857|14) (4591|13)
	(15|1)	(2622|37)
	(16|0)	(2479|24)
	(16|1)	(1987|36)
	(16|2)	(668|40) (4278|24)
	(16|3)	(1426|29)
	(17|0)	(2889|51)
	(17|1)	(186|19) (3355|20)
	(17|2)	(2245|29)
	(17|3)	(551|28) (3729|35)
	(18|0)	(2772|28)
	(18|1)	(1294|33) (4627|1)
	(19|0)	(886|13) (4199|19)
	(19|1)	(278|51) (3599|22)
	(19|2)	(2587|43)
	(19|3)	(759|7) (3736|41)
	(20|0)	(642|18) (4020|37)
	(20|1)	(673|34) (3707|22)
	(20|2)	(1134|19) (4003|39)
	(20|3)	(686|17) (3763|25)
	(21|0)	(2348|34)
	(21|1)	(1166|21) (4127|21)
	(21|2)	(867|26) (4103|26)
	(21|3)	(1048|25) (3953|35)
	(22|0)	(1908|27)
	(22|1)	(1815|12)
	(22|2)	(2309|27)

	(22|3)	(2035|29)
	(23|0)	(2234|39)
	(23|1)	(0|34) (3574|12)
	(23|2)	(2403|49)
	(23|3)	(349|1) (3230|48)
	(24|0)	(781|28) (3209|25)
	(24|1)	(2346|44)
	(25|0)	(364|13) (3707|27)
	(25|1)	(204|30) (2694|13)
	(25|2)	(2203|28)
	(25|3)	(2496|36)
	(26|0)	(2260|27)
	(26|1)	(2939|23)
	(27|0)	(441|48) (3609|29)
	(27|1)	(2228|16)
	(27|2)	(2399|19)
	(27|3)	(168|39) (3297|35)
	(28|0)	(1953|7)
	(28|1)	(2013|29)
	(29|0)	(1705|39)
	(29|1)	(1565|40)
	(30|0)	(2184|21)
	(30|1)	(941|29) (4330|39)

Table A.6. Resource Related Data - Problem 2

Part	Sub-process	Route	Operation	Minimum transport batch	Setup time	Unit execution time	Operation ready time	Machine type required	Other resources required (typelquantity)
0	0	0	0	16	7	1	0	6	
0	0	0	1	27	7	4	0	9	
0	0	0	2	21	3	4	0	7	(19l1) (22l1)
0	0	0	3	16	1	4	0	8	
0	0	0	4	46	4	1	0	10	(20l1) (21l1)
0	0	0	5	48	4	3	0	8	
0	1	1	6	27	7	2	0	10	(17l1) (22l1)
0	1	1	7	21	7	1	0	2	
0	1	2	8	16	6	1	0	1	(16l1)
0	1	2	9	42	5	3	0	4	
0	1	2	10	14	9	1	0	5	(25l1)
0	2	3	11	20	4	1	0	4	(22l1)
0	2	3	12	12	6	1	0	5	
0	2	3	13	13	6	2	0	9	(16l1)
1	3	4	14	19	5	1	0	10	(11l1) (18l1) (24l1)
1	3	4	15	16	7	3	0	6	(17l1)
1	4	5	16	15	3	3	0	8	(19l1)
1	4	5	17	34	7	3	0	5	(17l1) (25l1)
1	4	5	18	12	6	3	0	3	
1	4	5	19	11	8	5	0	6	(27l1)
1	4	5	20	17	4	2	0	8	
1	4	6	21	23	4	3	0	0	
1	4	6	22	17	5	1	0	10	(22l1)
1	5	7	23	11	5	1	0	7	(13l1) (21l1)
1	5	7	24	21	3	1	0	10	
1	5	8	25	11	7	1	0	6	(28l1)
1	5	8	26	12	3	1	0	7	(11l1) (25l1)
2	6	9	27	6	7	4	0	6	
2	6	9	28	6	3	4	0	4	(13l1) (16l1) (23l1)
2	6	9	29	26	5	3	0	8	(13l2)
2	6	9	30	12	7	1	0	6	(11l1)
2	7	10	31	20	6	4	0	9	(20l1) (25l1)
2	7	10	32	9	3	1	0	7	(18l1) (27l1)
2	7	10	33	21	7	5	0	10	(25l1)
2	7	10	34	7	5	2	0	9	
2	7	10	35	20	8	1	0	7	
2	8	11	36	6	4	5	0	10	
2	8	11	37	10	6	1	0	4	
2	8	11	38	24	8	1	0	5	(23l1)
2	8	11	39	8	4	1	0	9	(21l1)
2	8	11	40	11	3	2	0	6	(16l1)
2	8	11	41	13	4	4	0	3	
2	9	12	42	19	4	4	0	7	(24l1)
2	9	12	43	10	4	2	0	2	

2	9	12	44	12	4	3	0	0	(16|2) (18|1)
2	10	13	45	6	3	4	0	4	
2	10	13	46	15	4	2	0	9	(13|1) (25|1)
2	10	14	47	6	0	2	0	9	(17|1) (20|1)
2	10	14	48	7	3	4	0	2	(20|1) (22|1)
2	10	14	49	9	5	2	0	1	
2	10	14	50	9	5	3	0	3	
3	11	15	51	3	5	2	0	3	
3	11	15	52	10	7	4	13	10	(19|1)
3	11	16	53	9	3	1	0	7	
3	12	17	54	3	3	1	0	5	(17|1)
3	12	17	55	4	7	2	0	7	(30|1)
3	12	17	56	6	2	1	0	4	(19|1) (29|1)
3	13	18	57	3	2	3	0	1	(16|2) (20|1)
3	13	18	58	3	1	1	0	3	
3	13	18	59	7	8	1	0	4	(16|1) (22|1)
3	13	18	60	4	5	1	0	6	(27|1)
3	13	18	61	3	6	2	0	9	(22|1)
3	14	19	62	4	3	1	0	0	
3	14	19	63	7	6	3	0	3	
3	14	19	64	8	3	4	0	6	(23|1)
3	14	19	65	6	0	1	0	0	(16|1) (28|1)
3	14	19	66	12	2	3	0	4	
3	14	20	67	4	6	2	0	8	(21|1) (29|1)
3	15	21	68	4	7	3	0	10	
3	15	21	69	3	7	6	0	4	(16|1)
3	15	21	70	10	5	3	0	5	(16|2) (25|1)
3	15	22	71	3	5	3	0	1	(16|1)
3	15	22	72	4	3	2	0	10	
4	16	23	73	8	5	2	0	7	(16|1)
4	16	23	74	6	6	4	0	10	(20|1) (21|1) (22|1) (29|1)
4	16	23	75	3	3	5	0	1	(20|1) (22|1) (27|1)
4	17	24	76	6	5	3	0	7	(16|1) (25|1)
4	17	24	77	3	2	3	0	8	(17|1) (22|1) (27|1)
4	17	24	78	3	8	2	0	2	(16|1) (19|1) (20|1)
4	17	24	79	6	6	1	0	3	
4	17	25	80	6	6	6	0	8	
4	17	25	81	5	6	4	0	9	(16|1) (19|1)
4	17	25	82	7	5	5	0	10	
4	17	25	83	5	3	4	0	9	(20|1) (25|1)
4	17	25	84	4	5	4	0	10	(17|1)
4	18	26	85	13	7	4	0	6	(16|1)
4	18	26	86	3	7	1	0	10	(21|1)
4	18	26	87	5	7	1	0	5	(11|1) (13|1)
4	18	26	88	4	4	1	0	0	(20|2)
4	18	26	89	3	7	3	0	5	
4	19	27	90	9	4	2	0	6	(22|1) (29|1)

4	19	27	91	4	4	3	0	3	(15|1) (21|1)
4	19	27	92	9	5	3	0	10	(16|1) (17|1) (19|1) (23|1)
4	19	28	93	6	2	1	0	10	(16|1)
4	19	28	94	4	4	4	0	8	
4	20	29	95	8	5	5	0	7	(22|1)
4	20	29	96	10	2	2	0	5	(12|1)
4	20	29	97	5	3	4	0	10	(12|1) (15|1)
4	20	30	98	4	4	3	0	7	(17|1) (21|1)
4	20	30	99	4	4	3	0	2	
4	20	30	100	4	8	4	0	10	(16|1)
4	20	30	101	6	6	5	0	5	(25|1)
4	20	30	102	4	4	2	0	3	
4	20	30	103	3	6	4	0	10	(20|1)
5	21	31	104	7	2	2	0	7	(22|1)
5	21	31	105	4	3	1	0	3	
5	21	31	106	3	7	2	0	6	(16|1) (21|1) (22|1)
5	21	31	107	3	2	4	0	4	
5	21	31	108	4	7	5	0	5	(12|1) (22|1)
5	21	32	109	6	4	4	0	7	(13|1)
5	21	32	110	3	8	3	0	10	(16|1)
5	21	32	111	4	6	4	0	4	(23|2)
5	21	32	112	3	7	3	0	7	
5	21	32	113	3	3	1	0	4	
5	22	33	114	3	6	4	0	8	(12|1) (16|1) (27|1)
5	22	33	115	5	3	4	0	4	
5	22	33	116	8	0	2	0	5	(16|1)
5	22	33	117	9	3	3	0	10	(16|2)
6	23	34	118	20	6	4	0	10	(17|1) (22|1) (25|1)
6	23	34	119	6	6	2	0	6	(21|1) (23|2)
6	23	35	120	7	5	3	0	5	
6	23	35	121	12	1	3	0	10	(22|2)
6	23	35	122	10	7	2	0	3	
6	23	35	123	11	6	2	0	8	(16|2) (17|1) (22|1)
6	24	36	124	6	7	3	0	10	(22|1)
6	24	36	125	6	5	4	0	7	(16|1) (25|1)
6	24	36	126	9	0	3	0	10	(22|1) (23|1)
6	24	36	127	7	3	3	0	8	
6	25	37	128	6	4	1	0	9	(26|1)
6	25	37	129	6	4	1	0	3	
6	25	37	130	5	7	3	0	10	(12|1) (22|1)
6	25	37	131	5	4	3	0	8	
6	25	37	132	5	10	1	0	7	(25|1)
6	26	38	133	14	4	1	0	9	(21|1)
7	27	39	134	10	6	2	0	4	(12|1) (13|1)
7	28	40	135	5	5	4	0	7	
7	28	40	136	6	7	1	0	10	
7	28	40	137	6	3	2	0	3	

7	28	40	138	18	5	6	0	9	
7	28	40	139	8	3	4	0	8	(26|1)
7	28	40	140	8	9	1	0	2	
7	28	41	141	11	6	2	0	9	(22|1)
7	28	41	142	11	6	3	0	3	(21|2)
7	28	41	143	5	5	3	0	8	(16|1)
8	29	42	144	3	1	3	0	7	(21|1)
8	29	42	145	6	5	3	0	10	(12|1) (26|1)
8	30	43	146	3	4	1	0	8	
8	30	43	147	3	5	4	0	7	(20|2) (23|1) (25|1)
8	30	43	148	3	4	2	0	10	(11|1)
8	30	43	149	4	6	4	0	8	
8	30	43	150	5	5	1	0	7	(21|1)
8	30	44	151	7	6	5	0	6	(22|1)
8	30	44	152	8	6	1	0	0	
8	30	44	153	3	7	3	0	7	(20|1) (22|1)
8	31	45	154	4	6	2	0	5	(16|1)
8	31	45	155	3	4	3	0	8	
8	31	45	156	4	4	3	0	4	(20|1)
8	32	46	157	4	8	1	0	9	
8	32	46	158	2	3	4	0	10	
8	32	46	159	3	6	4	0	4	(25|2)
8	32	46	160	3	9	1	0	6	
8	32	46	161	3	6	4	0	10	(11|1) (20|1)
8	32	46	162	3	6	3	0	1	
9	33	47	163	3	2	1	0	0	
9	33	47	164	3	4	1	0	7	
9	33	47	165	3	7	4	0	4	(16|1)
9	33	47	166	3	6	3	0	8	
9	33	47	167	9	6	3	0	1	
9	34	48	168	2	1	2	0	6	(13|2)
9	34	48	169	2	7	2	0	5	(11|1) (22|2)
9	34	48	170	2	6	3	0	10	(20|1) (29|1)
9	34	48	171	3	6	3	0	9	(12|1)
9	35	49	172	2	6	4	0	0	(16|2) (20|1)
9	35	49	173	4	7	2	0	8	
9	35	49	174	4	5	2	0	9	(17|1)
9	35	49	175	3	4	2	0	4	(21|1) (25|1)
9	35	49	176	4	9	2	0	3	
10	36	50	177	4	4	3	0	4	
10	36	50	178	3	5	1	0	9	
10	37	51	179	10	8	2	0	5	(26|1)
10	37	51	180	10	0	2	0	8	
10	37	51	181	3	9	4	0	10	(11|1) (21|1)
10	38	52	182	8	5	1	0	3	
10	38	52	183	3	4	2	0	2	(25|1)
10	38	52	184	10	8	4	0	6	(16|1) (19|1) (21|1) (29|1)

10	38	52	185	5	4	2	0	3	
10	38	52	186	9	4	3	36	7	(15|2)
10	39	53	187	2	4	3	0	8	
10	39	53	188	2	5	3	0	3	
10	39	53	189	3	5	5	0	7	(19|1)
10	39	53	190	5	6	4	0	5	
11	40	54	191	4	7	4	0	10	(13|1)
11	40	54	192	7	4	1	0	2	(22|1)
11	40	54	193	3	8	2	0	0	(19|1)
11	40	54	194	9	5	3	0	9	(12|1) (25|1)
11	41	55	195	2	6	4	0	3	
11	41	55	196	3	7	4	0	9	(21|1) (25|1)
11	41	55	197	6	7	1	0	7	(20|1)
11	41	55	198	4	5	2	0	8	(11|1) (16|1) (24|1)
11	41	55	199	2	7	3	0	0	(20|1)
12	42	56	200	5	6	1	0	5	
12	42	56	201	10	3	1	0	2	
12	42	56	202	4	7	2	0	5	
12	42	56	203	5	2	3	0	9	
12	42	56	204	14	3	1	0	6	
12	43	57	205	5	3	3	0	9	(22|2)
12	43	57	206	7	4	2	0	5	
12	43	57	207	5	7	4	0	10	
12	43	57	208	7	7	5	0	8	
12	43	57	209	12	4	1	0	9	
13	44	58	210	3	0	1	0	7	(20|1)
13	44	58	211	4	6	4	0	5	
13	44	58	212	2	6	3	0	2	
13	44	58	213	2	4	1	0	8	
13	45	59	214	4	5	5	0	10	(25|1)
13	45	59	215	3	1	1	0	8	(20|2)
13	45	59	216	8	3	3	0	2	(20|1) (23|1)
13	45	59	217	8	4	3	0	9	(11|1)
14	46	60	218	4	7	1	0	8	
14	46	60	219	3	6	4	0	5	
14	46	60	220	2	7	1	0	9	
14	46	60	221	3	7	2	0	1	
14	46	60	222	2	7	1	0	2	
14	46	61	223	2	9	3	0	3	
14	47	62	224	4	3	1	0	8	(22|2) (25|1)
14	47	62	225	4	3	1	0	4	(11|1)
14	47	62	226	2	3	4	0	9	(17|1) (20|1)
14	47	62	227	3	6	2	0	1	(20|1)
14	48	63	228	2	8	2	0	6	
14	48	63	229	2	6	2	0	5	
14	48	63	230	4	3	3	0	10	(11|1) (25|2)
14	48	63	231	3	4	3	0	3	(27|1)
14	48	63	232	3	2	1	0	9	

14	49	64	233	4	6	1	0	1	(27|1) (29|1)
14	49	64	234	3	6	6	0	7	
14	49	64	235	3	7	3	0	5	(21|2)
14	49	64	236	2	6	3	0	10	
14	49	64	237	6	4	3	0	6	(29|1)
14	50	65	238	2	7	2	0	4	
14	50	65	239	2	5	3	0	3	
14	50	65	240	2	3	3	0	9	
15	51	66	241	14	4	4	0	5	
15	51	66	242	3	5	3	0	3	(16|1)
15	51	66	243	7	4	1	0	5	(14|1)
15	51	66	244	10	6	2	0	8	
16	52	67	245	3	3	4	0	5	(17|1) (20|1) (25|2) (29|1)
16	52	67	246	6	4	1	0	7	(29|2)
16	52	67	247	8	6	1	0	2	
16	52	67	248	5	2	2	0	8	(12|1) (25|1)
16	53	68	249	5	3	3	0	3	(25|1)
16	53	68	250	2	7	4	0	5	(16|1)
16	53	68	251	7	3	6	0	7	(19|1)
16	53	68	252	3	3	3	0	8	(20|2) (25|1)
16	53	68	253	6	2	3	0	10	
17	54	69	254	4	1	2	0	7	(22|1)
17	55	70	255	2	5	1	0	3	(16|1) (23|1)
17	55	70	256	2	5	4	0	8	(22|1)
17	56	71	257	5	6	2	0	4	(20|1) (25|1)
17	56	71	258	2	3	4	0	3	(19|1) (24|1)
17	56	71	259	2	5	1	0	10	(11|1) (17|1) (22|1)
17	56	71	260	2	7	3	0	3	(25|1)
17	56	71	261	5	5	3	0	1	(16|1)
17	56	71	262	2	8	2	0	8	(19|1) (20|1)
17	56	72	263	5	5	2	0	5	(20|1)
17	57	73	264	3	8	1	0	6	
17	57	73	265	6	5	5	0	2	
17	57	74	266	5	4	3	0	6	(22|1)
17	57	74	267	7	1	4	0	8	
17	57	74	268	4	3	1	0	2	
17	57	74	269	2	6	2	0	5	
18	58	75	270	16	4	2	0	5	(12|1)
18	58	75	271	9	3	3	0	10	(22|1) (25|2)
18	59	76	272	8	4	2	0	5	
18	59	76	273	15	7	1	0	1	(29|1)
18	59	76	274	11	5	2	0	9	
18	59	76	275	20	6	1	0	10	(16|1) (21|1)
18	59	76	276	7	4	2	0	7	
18	59	76	277	5	4	2	0	10	
19	60	77	278	13	9	1	0	9	(20|1) (25|1)
19	60	77	279	6	7	2	0	6	(13|1) (20|1) (21|1)

19	60	77	280	4	2	1	0	5	
19	60	77	281	4	3	3	0	10	(13|1) (14|1) (27|1)
19	60	77	282	17	8	6	0	9	
19	60	77	283	19	4	2	0	6	(27|1) (29|1)
19	61	78	284	7	8	4	0	5	(26|1)
19	61	78	285	6	6	1	0	4	
19	61	78	286	5	0	1	0	3	(25|1)
20	62	79	287	10	6	1	0	4	(22|1)
20	62	79	288	4	6	1	0	9	
20	63	80	289	9	7	3	0	6	(13|1) (19|1) (25|1)
20	63	80	290	5	3	4	0	8	(16|1)
20	63	80	291	7	1	4	0	5	(17|2) (22|1)
20	64	81	292	19	6	2	0	7	(22|1)
20	65	82	293	12	6	4	0	3	
20	65	82	294	5	6	5	0	7	(19|1)
21	66	83	295	3	3	3	0	6	(21|1)
21	66	83	296	3	5	3	0	9	
21	66	83	297	3	5	1	0	0	(27|1)
21	66	83	298	4	7	1	0	9	
21	66	83	299	3	6	2	0	6	(18|1) (25|1)
21	66	84	300	3	6	1	0	1	
21	66	84	301	9	8	3	0	0	(16|1)
21	66	84	302	10	7	6	0	2	(11|1) (25|1)
21	66	84	303	2	5	3	0	1	
21	66	84	304	2	6	3	0	0	(13|2)
21	66	84	305	4	4	4	0	10	(22|1)
21	67	85	306	8	4	4	0	5	(21|1)
21	67	85	307	3	7	6	0	8	
21	67	85	308	2	5	2	0	10	(17|1) (19|1) (22|1)
21	67	85	309	3	10	4	0	7	(22|1)
21	67	85	310	5	5	2	0	10	(26|1)
21	67	85	311	10	6	1	0	8	
21	67	86	312	2	2	1	0	8	
22	68	87	313	3	5	5	0	9	(17|1) (27|1)
22	68	87	314	4	9	4	0	6	(12|1)
22	68	87	315	3	2	3	0	5	
22	68	87	316	3	8	1	0	9	(17|1)
22	69	88	317	7	4	4	0	0	(16|1) (20|1) (27|2)
22	69	88	318	11	7	4	0	4	
22	69	88	319	6	6	3	0	10	(20|1) (25|1)
22	69	88	320	7	1	1	0	4	
22	70	89	321	3	7	3	0	8	
22	70	89	322	6	6	2	0	7	
22	70	89	323	11	5	3	0	3	(25|1)
22	70	90	324	4	2	4	0	7	(22|1)
22	70	90	325	3	5	3	0	10	
22	70	90	326	3	5	3	0	8	(21|1)
22	70	90	327	6	9	3	0	4	(17|1)

22	70	90	328	9	6	4	0	3	(16|1) (22|1)
22	71	91	329	5	6	1	0	10	(20|1) (22|1)
22	72	92	330	12	0	3	0	4	(17|1)
22	72	92	331	3	6	4	0	9	
22	72	92	332	9	5	1	0	10	(16|1) (20|1) (25|1) (27|1)
22	72	92	333	11	3	3	0	2	(16|1) (20|2)
22	72	92	334	6	9	2	0	4	
22	72	92	335	4	4	1	0	9	
23	73	93	336	4	8	1	0	3	
23	73	93	337	11	6	2	0	5	
23	73	93	338	6	4	4	0	3	(25|1)
24	74	94	339	15	6	4	0	10	(14|1) (17|1) (20|1) (29|1)
24	74	94	340	12	6	3	0	6	
24	74	94	341	27	4	6	0	3	
24	74	94	342	28	4	2	0	8	(17|1)
24	74	94	343	15	6	5	0	4	(21|1)
25	75	95	344	5	6	1	0	1	(29|1)
25	75	96	345	7	4	3	0	0	(16|1)
25	75	96	346	4	5	4	0	7	
25	75	96	347	4	8	1	0	5	(11|1) (13|1)
25	75	96	348	5	7	5	0	3	(20|1)
25	76	97	349	4	4	2	0	4	(27|1)
25	76	97	350	7	4	4	0	3	
25	77	98	351	4	6	5	0	6	
25	77	98	352	6	5	3	0	7	
25	77	98	353	4	6	1	0	2	(22|1)
26	78	99	354	3	3	3	0	8	(23|1)
27	79	100	355	3	3	1	0	4	(11|1) (19|1) (27|1)
27	79	100	356	5	6	2	0	1	(27|1)
27	79	100	357	2	6	3	0	7	(16|1)
27	79	100	358	2	7	2	0	8	(16|1)
28	80	101	359	7	3	1	0	1	
28	80	101	360	2	7	3	0	8	
28	80	101	361	3	8	3	0	4	
28	80	101	362	4	1	4	0	0	(20|2) (22|1)
28	80	101	363	2	1	3	0	5	
29	81	102	364	3	7	3	0	8	(16|1) (27|1)
29	81	102	365	4	8	3	0	7	
29	81	102	366	6	7	3	0	8	(21|1) (25|1)
29	81	102	367	3	8	2	0	6	(22|1)
29	82	103	368	2	4	2	0	10	
29	82	103	369	2	6	4	0	9	(12|1) (22|1)
29	82	103	370	4	6	1	0	3	
29	82	103	371	3	6	2	0	8	
29	83	104	372	6	1	3	0	9	(16|1) (22|1)
29	83	104	373	3	2	1	0	4	(20|1)

29	83	104	374	3	6	1	0	5	(15|1) (17|1)
29	83	104	375	5	5	2	0	8	(11|1) (22|2)
30	84	105	376	12	6	3	0	6	(19|1) (21|1) (28|1)
30	84	105	377	8	7	1	0	4	(21|1) (29|1)
30	84	105	378	6	3	1	0	5	(20|1) (22|1)
30	84	106	379	5	5	3	0	1	(13|1) (17|1) (20|1) (23|1)
30	84	106	380	6	4	3	0	7	
30	84	106	381	4	5	4	0	5	(20|1)
30	84	106	382	5	3	3	0	9	(16|1)
30	84	106	383	11	10	2	0	10	
30	84	106	384	5	9	3	0	6	(13|1) (16|1) (21|1) (25|1)
30	85	107	385	8	4	3	0	7	
30	85	107	386	4	6	2	0	1	
30	85	107	387	12	6	2	0	3	
30	85	107	388	5	2	4	0	0	
30	85	108	389	9	4	1	0	8	
30	85	108	390	8	5	2	0	3	(20|1)
30	85	108	391	13	5	4	0	10	
30	86	109	392	4	8	4	0	4	
30	86	109	393	5	2	2	0	6	(11|1) (30|1)
30	86	109	394	6	2	4	0	4	(16|1)
30	86	109	395	5	5	1	0	8	(20|2)
30	86	109	396	8	4	3	0	3	(22|2)
30	86	110	397	7	5	1	0	6	
30	86	110	398	5	4	5	0	9	(16|1) (18|1)
30	86	110	399	8	6	3	0	7	(12|1) (17|1) (23|1)
30	86	110	400	7	6	4	0	6	(22|1)
30	87	111	401	5	4	3	0	4	(20|1)
30	87	111	402	5	6	2	0	5	(22|1)
30	87	111	403	6	3	5	0	8	(22|1) (25|1)
30	87	112	404	5	1	4	0	4	(20|1)
30	87	112	405	6	4	1	0	8	(21|1)
30	87	112	406	4	4	3	0	3	(21|1)
30	87	112	407	7	5	1	0	1	(17|1) (21|1)
31	88	113	408	2	6	1	0	1	
31	88	113	409	9	4	1	0	7	(23|1)
31	88	113	410	2	5	1	0	3	(22|1)
31	89	114	411	3	5	4	0	7	(13|1)
31	89	114	412	4	4	2	0	1	
31	89	114	413	8	5	3	0	4	(25|1)
31	90	115	414	6	0	4	0	9	
31	90	115	415	8	4	2	0	6	(22|1)
31	90	115	416	3	4	1	0	5	(28|1)
31	90	115	417	5	6	5	0	0	
31	90	115	418	3	5	2	0	2	(22|2)
31	90	115	419	3	4	2	0	10	(11|2) (16|1)

31	91	116	420	3	9	3	0	3	(19|1)
31	91	116	421	2	8	3	0	7	(19|1)
31	91	116	422	2	9	1	0	5	
31	91	116	423	2	4	3	0	4	(16|1)
31	91	116	424	5	3	3	0	6	(16|1) (22|1) (25|1)
31	91	117	425	5	4	5	0	3	
31	91	117	426	2	9	4	0	9	
31	91	117	427	6	7	2	0	8	(20|1) (22|2) (25|1)
31	92	118	428	3	7	1	0	7	
31	92	118	429	2	6	1	0	5	
31	92	118	430	4	5	2	0	4	(16|1)
31	92	118	431	4	7	4	0	8	(29|1)
31	92	118	432	2	7	1	0	6	
31	92	119	433	3	6	4	0	10	(17|1)
31	92	119	434	4	6	2	0	7	
31	92	119	435	2	8	4	0	10	(11|1)
31	92	119	436	2	8	4	0	5	(23|1)
32	93	120	437	6	6	1	0	7	
32	93	120	438	5	6	1	0	9	(13|1) (20|1)
32	93	120	439	4	7	3	0	6	(22|1) (25|1)
32	94	121	440	6	4	2	0	3	(20|1) (30|1)
32	94	121	441	5	7	1	0	9	
32	94	122	442	4	8	4	0	3	(13|1) (20|2) (23|1)
32	94	122	443	13	4	4	0	4	
32	95	123	444	6	6	4	0	3	(16|1)
32	95	123	445	6	7	2	0	5	
32	95	123	446	6	6	3	0	2	(25|1)
32	95	124	447	5	6	1	0	6	(16|1) (20|1) (21|1)
32	96	125	448	7	7	4	0	10	
32	96	125	449	4	4	3	0	7	
33	97	126	450	3	5	2	0	3	
33	97	126	451	2	6	3	0	8	(13|1) (18|1) (22|1)
33	98	127	452	4	5	4	0	7	(16|1)
33	98	127	453	2	8	3	0	3	
33	98	127	454	3	4	1	0	4	
33	98	127	455	2	3	3	0	5	(20|1) (28|1)
33	99	128	456	2	1	3	0	10	(12|1)
33	99	128	457	7	7	3	0	8	(16|1) (20|1)
33	99	128	458	3	6	3	0	5	
33	99	128	459	8	5	1	29	10	(12|1) (16|1)
33	99	128	460	5	8	1	0	8	
33	99	129	461	3	3	1	0	4	(20|2) (25|2)
33	99	129	462	2	7	3	0	9	(13|1) (19|1) (20|1)
33	99	129	463	5	5	4	0	3	
33	99	129	464	3	10	4	0	8	(20|1) (25|1)
33	99	129	465	4	7	4	0	7	(13|1)
33	99	129	466	2	9	1	0	4	(12|1) (16|1)
33	100	130	467	3	4	5	0	6	(22|2)

34	101	131	468	4	4	2	0	3	
35	102	132	469	4	7	3	0	7	(20|1) (22|1)
35	102	132	470	5	4	3	0	10	
35	102	132	471	4	0	4	0	3	(16|1) (21|1) (25|1)
35	103	133	472	5	8	2	0	7	
35	103	133	473	6	2	3	0	6	(20|1) (30|1)
35	103	133	474	5	5	5	0	3	
35	103	133	475	2	6	3	0	8	
35	103	133	476	3	8	3	0	0	(25|1) (28|1)
35	104	134	477	2	4	6	0	6	
35	104	134	478	2	9	4	0	9	
35	104	134	479	2	7	4	0	4	(17|1) (29|1)
35	104	134	480	3	4	3	0	1	
35	104	134	481	3	1	3	0	8	
35	105	135	482	2	6	5	0	5	
35	106	136	483	3	8	4	0	1	(22|2)
35	106	136	484	4	5	4	0	6	(19|1)
35	106	136	485	2	6	5	0	4	(16|1)
36	107	137	486	7	4	4	0	10	(20|1)
36	107	137	487	6	1	3	0	6	
36	107	137	488	7	3	3	0	3	
36	107	138	489	5	7	2	0	6	
36	108	139	490	5	4	1	0	9	(22|1)
36	108	139	491	4	6	4	0	8	
36	108	139	492	6	8	3	0	10	(21|1)
36	108	139	493	4	3	2	0	5	(14|1)
36	108	139	494	5	8	1	0	10	(12|1)
36	108	139	495	8	2	2	0	8	(22|2) (25|1)
36	109	140	496	5	1	2	0	7	(15|1) (20|1)
36	109	140	497	6	5	1	0	4	(22|1)
36	109	140	498	7	3	1	0	5	(16|1) (25|1)
36	109	140	499	4	5	5	0	0	(20|1)
37	110	141	500	4	4	4	0	9	
37	110	141	501	7	9	4	0	3	(19|1)
37	110	141	502	3	4	3	0	8	(17|1)
37	111	142	503	4	8	3	0	3	(16|1) (23|1) (27|1)
37	111	142	504	3	3	3	0	0	
37	111	142	505	4	6	2	0	6	
37	111	142	506	2	3	3	0	3	
37	111	142	507	3	6	2	0	6	(17|1)
37	111	142	508	3	6	1	0	7	

Table A.7. Routing Structure and Operation Related Data - Problem 2

Operation	Setup times (preceding operation | setup)
7	(43|4) (78|4) (140|2) (192|6) (201|4) (212|5) (222|2) (265|4) (302|4) (333|1) (353|6) (418|6)
21	(62|3) (65|5) (88|5) (163|3) (172|3) (193|6) (199|5) (301|2) (304|5) (317|3) (362|5) (388|8) (417|2) (476|5) (499|5)
43	(48|5) (78|3) (99|3) (140|8) (192|6) (201|7) (222|3) (265|0) (302|9) (333|3) (418|6) (446|6)
48	(7|5) (78|5) (140|7) (192|7) (201|7) (212|4) (222|5) (302|4) (333|7) (353|5) (418|5) (446|6)
57	(71|3) (75|7) (162|7) (227|9) (233|6) (261|10) (273|5) (303|5) (356|9) (379|7) (407|7) (412|7)
62	(21|2) (65|4) (88|4) (152|6) (163|1) (172|6) (199|6) (301|6) (304|4) (317|8) (362|7) (388|7) (417|6) (476|5) (499|5)
65	(21|5) (62|3) (88|4) (152|8) (163|8) (193|4) (199|5) (304|7) (317|4) (362|6) (388|6) (417|8) (476|6) (499|6)
71	(57|4) (75|3) (162|4) (227|2) (261|8) (273|3) (300|3) (344|4) (356|2) (359|4) (407|4) (408|6) (412|6)
75	(57|4) (71|6) (162|2) (167|6) (221|8) (227|5) (233|6) (273|2) (344|5) (356|5) (359|6) (379|4) (407|6) (408|6) (412|8)
78	(7|4) (43|5) (48|4) (99|4) (212|7) (222|5) (265|5) (302|3) (333|3) (418|4)
88	(62|7) (65|3) (163|5) (172|5) (193|8) (199|3) (304|7) (317|5) (362|5) (388|7) (417|6) (476|2) (499|6)
99	(7|3) (43|7) (48|3) (78|2) (140|8) (201|5) (212|7) (222|5) (265|7) (302|7) (333|6) (353|8) (418|5) (446|6)
140	(43|6) (48|6) (78|6) (99|3) (192|2) (201|5) (302|6) (333|2) (353|4) (418|8) (446|4)
152	(21|8) (62|5) (65|3) (163|2) (172|2) (193|3) (199|1) (301|5) (304|4) (317|7) (362|5) (388|5) (476|4) (499|2)
162	(57|8) (71|5) (75|5) (221|3) (227|5) (233|4) (261|4) (273|3) (300|5) (303|5) (344|3) (356|2) (379|5) (412|5)
163	(21|10) (62|4) (65|4) (88|5) (152|5) (172|4) (193|5) (199|9) (301|4) (304|6) (317|5) (362|3) (388|6) (476|7) (499|8)
167	(71|9) (75|5) (162|2) (221|5) (233|1) (300|8) (303|3) (356|3) (359|3) (379|5) (407|1) (408|4) (412|4)
172	(21|2) (62|5) (88|2) (152|1) (163|2) (193|2) (199|8) (301|5) (304|1) (317|7) (362|7) (476|7) (499|7)
192	(7|7) (43|6) (78|3) (99|7) (140|5) (201|3) (222|8) (265|3) (302|6) (333|0) (353|3) (418|3) (446|3)

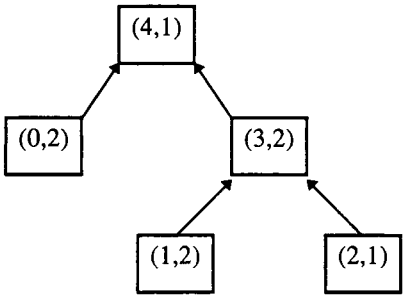
193	(21|4) (62|6) (163|4) (172|6) (199|1) (301|5) (317|2) (362|5) (388|4) (417|9) (476|2) (499|4)
199	(21|4) (62|5) (65|4) (88|4) (152|4) (163|6) (172|4) (193|5) (301|4) (304|3) (317|4) (362|6) (417|5) (476|5) (499|4)
201	(7|4) (78|1) (99|6) (140|1) (212|7) (222|5) (265|8) (302|6) (353|4) (418|5) (446|2)
212	(7|3) (43|5) (48|3) (78|7) (99|2) (140|8) (192|5) (201|1) (222|5) (302|8) (333|4) (418|4)
221	(57|4) (71|8) (75|2) (162|3) (167|0) (227|4) (233|3) (261|5) (300|5) (344|6) (356|8) (379|5) (407|6) (408|4) (412|4)
222	(7|6) (43|6) (48|6) (78|3) (99|5) (140|3) (192|5) (201|9) (212|6) (265|5) (302|3) (353|6) (446|3)
227	(57|3) (71|4) (75|1) (162|7) (167|5) (233|7) (261|2) (300|8) (303|4) (344|0) (356|2) (359|4) (379|5) (408|7)
233	(57|2) (71|5) (162|4) (167|7) (221|0) (227|8) (261|4) (300|5) (303|8) (344|5) (407|7) (408|3) (412|7)
261	(57|8) (71|7) (75|7) (167|3) (221|4) (227|4) (233|6) (273|1) (300|6) (303|6) (359|2) (379|4) (408|6) (412|7)
265	(7|4) (43|4) (48|7) (78|7) (99|4) (140|6) (192|1) (201|4) (222|4) (333|6) (353|7) (418|9) (446|6)
273	(57|6) (71|3) (75|6) (162|5) (221|2) (227|3) (261|4) (300|6) (303|6) (356|5) (359|4) (407|4) (408|5) (412|5)
300	(57|7) (71|5) (75|2) (162|8) (221|9) (227|5) (261|2) (303|8) (356|5) (359|5) (379|5) (407|5) (412|5)
301	(21|0) (62|5) (88|4) (152|5) (172|4) (193|3) (199|9) (304|10) (317|3) (362|7) (388|7) (417|7) (499|4)
302	(7|8) (43|4) (48|6) (78|3) (99|8) (140|4) (212|8) (265|3) (333|5) (353|5) (418|4) (446|5)
303	(57|3) (71|6) (75|4) (162|2) (167|3) (227|6) (233|6) (273|6) (300|7) (344|4) (356|10) (359|6) (407|8) (408|7) (412|7)
304	(62|10) (65|3) (88|4) (152|5) (163|4) (172|5) (193|7) (199|5) (301|4) (317|8) (362|7) (388|5) (499|5)
317	(21|5) (62|9) (65|3) (88|8) (152|9) (163|6) (199|7) (301|7) (304|6) (362|7) (388|6) (417|5) (476|5) (499|9)
333	(7|6) (43|7) (48|5) (140|6) (192|4) (201|5) (212|0) (222|5) (302|6) (353|6) (418|2) (446|1)
344	(71|7) (75|8) (167|4) (221|5) (227|3) (233|5) (261|4) (273|7) (300|3) (303|8) (356|4) (359|2) (379|7) (407|2) (408|3) (412|8)
353	(43|9) (48|2) (78|4) (99|4) (140|7) (192|4) (201|10) (212|5) (265|3) (302|7) (418|5) (446|2)
356	(57|7) (71|7) (75|5) (162|8) (221|4) (227|5) (233|5) (261|7) (273|8) (300|5) (303|3) (359|8)

	(379|4) (407|5) (408|5) (412|5)
359	(71|4) (75|4) (162|4) (227|5) (233|2) (261|1) (273|5) (300|2) (356|2) (407|5) (408|2) (412|1)
362	(21|3) (62|5) (65|4) (88|5) (152|5) (163|2) (193|6) (199|3) (301|6) (304|4) (388|4) (417|4) (476|4) (499|3)
379	(57|7) (71|2) (162|4) (167|7) (221|3) (227|3) (233|3) (261|8) (273|6) (300|7) (303|3) (344|8) (359|8) (407|6) (412|3)
388	(21|5) (65|7) (88|7) (152|4) (163|4) (172|6) (193|3) (304|6) (317|8) (476|6) (499|5)
407	(57|7) (71|4) (162|3) (221|7) (227|8) (261|3) (300|2) (303|3) (344|6) (356|8) (359|2) (379|4)
408	(57|5) (71|4) (75|5) (162|7) (167|0) (227|4) (233|9) (261|9) (300|3) (303|9) (344|3) (356|10) (359|4) (379|7) (407|4)
412	(57|7) (71|7) (75|0) (162|6) (221|8) (233|3) (261|0) (273|6) (300|8) (303|3) (356|5) (359|8) (379|0)
417	(21|3) (62|8) (65|0) (88|5) (152|4) (163|5) (172|7) (193|1) (199|7) (304|7) (317|4) (362|7) (388|5) (476|9) (499|7)
418	(7|4) (48|6) (78|4) (99|4) (140|4) (192|6) (201|6) (212|7) (222|6) (265|6) (302|8) (333|3) (353|7) (446|4)
446	(7|5) (43|4) (48|1) (78|4) (99|8) (140|7) (192|4) (201|5) (212|4) (222|10) (265|8) (302|5) (333|3) (353|3) (418|4)
476	(62|4) (65|6) (88|4) (163|5) (172|4) (193|7) (301|7) (304|6) (317|6) (362|2) (499|4)
499	(21|6) (62|9) (65|3) (88|6) (152|4) (199|2) (304|3) (317|4) (362|6) (417|4) (476|7)

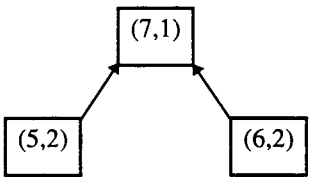
Table A.8. Operations with Sequence Dependent Setup Times - Problem 2

Problem 3: The following set of tables and figures specify problem 3.

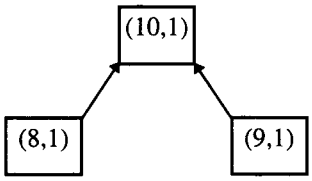
Job 0:



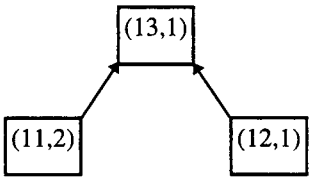
Job 1:



Job 2:

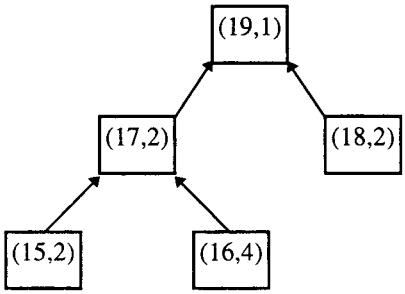


Job 3:



Job 4:

Job 5:



Job 6:

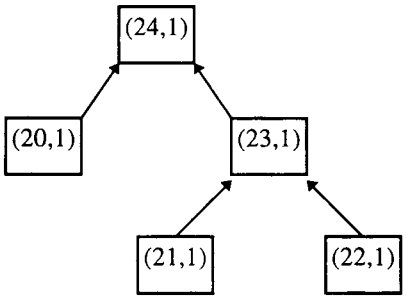


Figure A.3. BOMs - Jobs from Problem 3

Job	batch size	due date	ready time
0	10	1192	0
1	10	2596	0
2	9	1948	0
3	11	2992	0
4	11	1984	0
5	10	2452	0
6	12	3532	0

Table A.9. Job Related Dada - Problem 2

Resource class	Resource (type|unit)	Scheduled breaks (start time|duration)
Machines	(0|0)	(912|18) (2830|35)
	(1|0)	(312|29) (2121|17)
	(2|0)	(326|28) (2334|21)
	(2|1)	(388|31) (2379|45)
	(2|2)	(405|42) (2607|40)
	(2|3)	(1044|37) (2801|23)
	(3|0)	(21|16) (1817|39)
	(4|0)	(266|46) (2152|42)
	(4|1)	(1843|15)
	(4|2)	(1217|19)
	(5|0)	(1195|22)
	(5|1)	(1525|32)
	(5|2)	(1901|24)
	(6|0)	(823|33) (3056|30)
	(7|0)	(196|12) (2328|35)
	(7|1)	(1049|40) (3149|39)
Other resources	(8|0)	(342|34) (2456|28)
	(8|1)	(530|30) (2340|41)
	(8|2)	(1159|40)
	(8|3)	(0|24) (2044|33)

(9|0)	(1916|26)
(9|1)	(418|39) (2297|26)
(9|2)	(862|39) (2501|18)
(9|3)	(808|43) (2851|22)
(10|0)	(365|20) (2525|30)
(10|1)	(1171|25)
(10|2)	(963|32) (3155|19)
(10|3)	(1824|27)
(11|0)	(1293|40)
(11|1)	(526|17) (2243|24)
(12|0)	(1555|27)
(12|1)	(1431|40)
(12|2)	(2037|22)
(12|3)	(1373|27)
(13|0)	(340|23) (2243|29)
(13|1)	(873|25) (3198|21)
(13|2)	(525|44) (2429|16)
(13|3)	(828|12) (2880|32)
(14|0)	(733|27) (2580|37)
(14|1)	(998|45) (2683|23)
(14|2)	(1654|40)
(14|3)	(1128|38) (2946|22)
(15|0)	(1810|14)
(15|1)	(1224|38)
(15|2)	(505|16) (2361|33)
(15|3)	(745|21) (2786|38)
(16|0)	(1352|34)
(16|1)	(2093|34)
(16|2)	(1365|35)
(16|3)	(73|20) (2033|38)
(17|0)	(1905|29)
(17|1)	(1043|27) (2870|36)

	(17|2)	(946|35) (2701|16)
	(17|3)	(1509|29)
	(18|0)	(1656|47)
	(18|1)	(539|12) (2411|21)
	(19|0)	(1155|44)
	(19|1)	(1104|42) (3286|40)
	(19|2)	(1302|23)
	(19|3)	(1693|34)
	(20|0)	(300|18) (2298|37)
	(20|1)	(84|18) (1742|28)
	(20|2)	(19|10) (2209|33)
	(20|3)	(601|15) (2436|54)
	(21|0)	(42|41) (2303|12)
	(21|1)	(1260|28)
	(21|2)	(1411|45)
	(21|3)	(443|21) (2424|21)
	(22|0)	(1128|16) (3304|26)
	(22|1)	(1862|41)

Table A.10. Resource Related Data - Problem 3

Part	Sub-process	Route	Operation	Minimum transport batch	Setup time	Unit execution time	Operation ready time	Machine type required	Other resources required (type|quantity)
0	0	0	0	5	6	3	0	7	
0	1	1	1	5	6	3	0	4	(19|1)
0	1	1	2	4	6	3	0	3	(17|1)
0	1	2	3	10	8	2	0	5	(10|1)
0	1	2	4	11	5	1	0	7	
0	1	2	5	6	4	1	0	3	(9|1) (12|2)
0	2	3	6	6	3	4	0	7	(10|1) (13|1)
0	2	3	7	16	3	2	0	3	
0	2	3	8	19	4	2	0	5	(8|1) (13|1) (14|1)
0	2	3	9	5	4	4	0	7	(21|1) (22|1)
1	3	4	10	15	6	4	0	3	(9|1)
1	3	4	11	4	6	3	0	2	(19|1)
1	3	4	12	6	4	4	0	1	
1	3	4	13	5	5	5	0	2	(20|1)
1	3	4	14	7	7	3	0	5	(9|1) (21|1)
1	4	5	15	20	3	3	0	7	(8|1)
1	4	5	16	6	5	6	0	6	(15|1) (21|1)
1	4	6	17	4	5	2	0	3	(20|1)
1	5	7	18	4	3	4	0	7	(10|1)
1	5	7	19	6	7	1	0	4	
1	5	7	20	6	3	2	0	2	
1	5	7	21	7	8	6	0	7	
1	6	8	22	5	6	1	0	2	(16|1)
1	6	8	23	7	8	1	0	0	(21|1)
1	6	8	24	6	2	2	0	1	(19|1) (21|1)
1	6	8	25	17	7	6	0	0	
1	6	8	26	20	5	2	0	4	(14|1)
1	6	9	27	20	5	3	0	2	
1	6	9	28	5	4	3	0	4	(12|2)

1	7	10	29	19	4	3	0	6	(19|1)
2	8	11	30	4	4	6	0	3	(8|1)
2	8	11	31	4	5	6	0	5	
2	8	11	32	2	4	1	0	0	
2	8	11	33	6	8	1	0	5	(12|1) (13|1) (21|1)
2	9	12	34	3	2	2	0	0	(12|1) (20|1)
2	9	13	35	4	6	4	0	1	(8|1) (20|1)
2	9	13	36	2	7	2	0	0	(19|1)
2	9	13	37	3	8	1	0	2	(14|1) (20|1)
2	10	14	38	3	2	3	0	4	(13|1)
2	10	14	39	10	1	4	0	5	(9|1) (12|1)
2	10	14	40	3	3	1	0	7	(9|1) (21|2) (22|1)
3	11	15	41	5	5	4	0	2	(14|1) (16|1) (18|1)
3	11	15	42	10	5	1	0	5	(14|1) (19|1)
3	11	16	43	7	4	1	0	5	
3	11	16	44	13	6	2	0	4	
3	11	16	45	9	4	1	0	7	(15|1) (21|1)
3	11	16	46	5	5	4	0	5	
3	12	17	47	9	5	3	0	4	
3	12	18	48	5	5	5	0	4	(9|1) (13|1)
3	12	18	49	5	5	1	0	7	
3	12	18	50	9	3	3	0	5	(14|2) (19|1)
3	12	18	51	7	4	4	0	4	
3	12	18	52	7	6	1	0	5	(15|1) (20|1) (21|1)
3	13	19	53	13	5	4	0	2	(9|1) (12|1) (19|1) (21|1) (22|1)
3	13	19	54	4	4	2	0	5	(14|1)
3	14	20	55	5	3	1	0	3	(16|1) (21|1) (22|1)
3	14	20	56	4	5	1	0	5	
3	14	20	57	5	9	4	0	1	(10|1) (13|1)
3	14	20	58	7	8	3	0	6	
3	14	21	59	11	3	5	0	6	(10|1) (13|1) (14|1)

3	14	21	60	11	4	2	0	2	(15|1) (19|1)
3	14	21	61	19	2	4	0	5	(16|1) (20|1)
3	15	22	62	5	1	3	0	7	(14|1)
3	15	22	63	8	3	3	0	6	(12|1) (22|1)
3	15	23	64	5	5	3	0	2	(19|1)
3	15	23	65	9	5	2	0	1	(17|1) (19|1) (21|2)
3	15	23	66	5	5	4	0	6	
4	16	24	67	3	5	2	0	2	(8|1) (14|1) (20|1)
4	16	24	68	4	6	2	0	4	(20|1) (21|1)
4	16	25	69	4	1	3	0	0	
4	16	25	70	4	4	1	0	3	
4	16	25	71	6	5	3	0	5	
4	17	26	72	6	4	4	0	6	(12|1) (16|1)
4	17	26	73	2	3	6	0	0	(9|1) (21|1)
4	17	27	74	7	4	5	0	7	(8|1)
4	17	27	75	8	5	1	0	6	
4	17	27	76	7	6	1	0	2	(16|2) (19|1)
4	17	27	77	2	6	4	0	6	
4	18	28	78	4	3	2	0	5	(9|1)
4	18	28	79	4	7	2	0	4	(19|1)
4	19	29	80	3	4	1	0	3	(13|1)
4	19	29	81	3	5	2	0	6	(8|1)
4	19	29	82	2	4	4	0	2	
4	19	30	83	2	6	1	0	6	
4	19	30	84	4	4	2	0	2	(10|1)
4	19	30	85	4	5	5	0	7	(9|1)
5	20	31	86	4	10	1	0	6	(12|1) (20|1)
5	20	31	87	9	4	3	0	2	(8|1) (13|1)
5	20	32	88	10	3	2	0	0	(9|1)
5	20	32	89	20	1	2	0	2	(19|1)
5	20	32	90	9	3	5	0	7	(15|1)
5	21	33	91	11	5	5	0	4	(9|1) (14|1) (21|1)

5	21	33	92	7	4	1	0	7	(9|1) (14|1) (19|1)
5	21	33	93	4	0	5	0	3	(12|1)
6	22	34	94	7	9	1	0	3	
6	22	34	95	7	7	1	0	6	(13|1) (14|1) (21|1)
6	22	34	96	4	7	2	0	1	
6	23	35	97	5	4	1	0	4	
6	23	36	98	7	9	2	0	2	(15|1)
6	23	36	99	5	2	4	0	4	(10|2) (13|1)
6	23	36	100	6	9	2	0	7	
6	23	36	101	6	1	1	0	5	
6	24	37	102	8	6	1	0	6	(13|1) (21|1)
6	24	37	103	4	4	2	0	3	
6	24	37	104	6	6	5	0	2	(12|1) (20|1)
6	25	38	105	6	4	1	0	4	(15|1) (19|1)
6	25	38	106	4	0	3	0	7	(10|2) (21|1)
6	25	38	107	6	4	5	0	3	(8|1)
6	25	39	108	6	7	4	0	4	
6	25	39	109	10	6	3	0	2	(16|1) (21|1)
6	25	39	110	19	6	2	0	6	(15|1)
6	25	39	111	4	4	2	0	2	(15|1)
6	26	40	112	18	4	5	0	6	(9|1) (15|1)
6	26	40	113	7	3	4	0	1	(10|1)
6	26	40	114	10	4	3	0	4	
6	26	40	115	8	4	3	0	7	(13|1) (21|1)
6	26	40	116	7	4	2	0	4	(19|1)
7	27	41	117	5	8	1	0	6	
7	27	41	118	9	4	4	0	1	(8|1) (10|1) (13|1)
7	28	42	119	6	4	1	0	3	(9|1)
7	28	42	120	4	6	3	0	2	(12|1)
7	28	42	121	3	3	4	0	3	(14|2) (15|1)
7	28	42	122	5	7	5	0	6	
7	28	42	123	6	4	4	0	3	

7	29	43	124	3	4	1	0	2	
7	30	44	125	5	6	1	0	7	(14|1)
7	30	44	126	2	3	1	0	4	(10|1) (20|1)
7	30	44	127	4	8	5	0	6	(13|1) (20|1)
7	30	44	128	7	7	1	0	7	(8|1) (14|1)
7	30	44	129	5	5	3	0	2	(11|1)
8	31	45	130	3	5	3	0	7	(19|1)
8	31	45	131	4	9	3	0	2	
8	31	45	132	3	5	1	0	6	(10|1) (13|1) (19|1)
8	31	46	133	4	5	4	0	2	(12|1) (22|1)
8	31	46	134	4	7	1	0	1	
8	31	46	135	6	6	4	0	5	(16|1) (19|1)
9	32	47	136	6	5	3	0	7	(19|1) (21|1)
9	32	47	137	5	6	1	0	0	
9	33	48	138	4	6	4	0	6	(14|1) (20|1)
9	33	48	139	2	5	4	0	7	
9	33	48	140	7	7	4	0	1	
9	33	48	141	3	5	4	0	6	(14|2)
9	33	49	142	7	4	3	0	3	
9	33	49	143	3	6	1	0	2	(10|1)
9	33	49	144	8	7	4	0	3	(15|1)
9	33	49	145	3	4	3	0	7	(16|2)
9	34	50	146	4	5	5	0	4	
9	34	50	147	2	2	3	0	7	(13|1)
9	34	51	148	2	2	4	0	1	(12|1)
10	35	52	149	5	2	1	0	2	(21|1)
10	35	52	150	2	5	1	0	0	(9|1)
10	35	52	151	2	8	3	0	1	(19|2)
10	36	53	152	7	6	4	0	6	
10	36	53	153	6	2	2	0	7	
10	37	54	154	5	8	5	0	2	(9|1) (13|1)
10	37	54	155	3	3	2	0	3	(13|1) (20|1)

10	37	54	156	2	5	2	0	1	(10|1)
10	38	55	157	7	9	4	0	6	(12|2) (15|1) (20|1)
10	38	55	158	3	5	1	0	2	(16|2)
11	39	56	159	5	9	3	0	2	
11	39	56	160	6	4	3	0	1	(15|1)
11	39	56	161	7	4	3	0	3	
11	39	57	162	5	5	2	0	7	(8|1) (13|1)
11	40	58	163	12	4	3	0	4	(16|1) (21|1)
11	40	58	164	6	4	1	0	7	(19|1) (20|1)
11	40	58	165	4	6	4	0	2	
11	40	58	166	5	7	1	0	3	
11	41	59	167	11	3	4	0	5	(21|1)
11	41	59	168	9	7	6	0	6	(14|1)
11	41	59	169	14	9	3	0	7	(19|1)
11	41	59	170	8	5	2	0	2	(20|1)
12	42	60	171	7	2	1	0	7	(10|1)
13	43	61	172	2	2	3	0	3	(21|1)
13	43	61	173	4	10	5	0	7	(20|1)
13	43	61	174	4	0	1	0	4	(20|1)
13	43	61	175	4	6	1	0	2	
13	43	61	176	3	2	1	0	5	(14|1)
13	43	62	177	3	6	2	89	6	(8|1) (13|1) (18|1)
13	43	62	178	2	7	1	0	4	(15|1) (21|1)
13	43	62	179	4	4	4	0	7	(17|2)
13	43	62	180	4	4	1	0	4	(19|1)
14	44	63	181	4	9	4	0	2	
14	44	63	182	7	4	3	0	6	(10|1) (16|2)
14	44	63	183	3	7	2	0	5	(8|1) (12|1)
14	44	63	184	6	3	4	0	3	(10|1) (14|1) (20|1)
14	44	63	185	3	3	2	0	6	(16|1)
14	45	64	186	11	8	1	0	7	(9|1)
14	45	64	187	8	7	4	0	6	(13|1) (21|1)

14	45	64	188	2	8	5	0	3	
14	45	64	189	3	6	1	0	5	(8|1) (17|1)
14	45	64	190	7	2	4	0	0	(16|1)
14	46	65	191	7	2	1	0	2	(14|1)
14	46	65	192	7	5	2	0	5	(21|1)
15	47	66	193	5	4	4	0	2	
15	47	66	194	15	5	3	0	0	
15	47	66	195	9	4	3	0	5	
15	48	67	196	4	6	4	0	3	
15	49	68	197	4	6	4	0	6	
15	49	68	198	6	4	5	0	5	(8|1) (19|1)
15	49	69	199	6	3	1	0	6	
15	49	69	200	4	6	4	0	4	(10|2) (16|1)
15	49	69	201	6	6	2	0	6	(9|1)
15	50	70	202	5	4	5	0	7	(16|1) (21|1)
15	50	70	203	6	5	1	0	5	
15	50	70	204	8	3	3	0	7	
15	50	70	205	13	3	3	0	6	(20|1)
15	50	70	206	9	7	1	0	4	(10|2)
16	51	71	207	14	0	5	0	6	(20|1)
16	51	71	208	8	9	2	0	0	
16	51	71	209	15	8	2	0	4	(21|1)
16	51	72	210	22	6	1	0	6	
16	51	72	211	12	5	3	0	2	
16	51	72	212	21	7	2	0	6	(12|2) (16|1)
16	51	72	213	9	6	1	0	7	(19|1)
16	51	72	214	10	3	3	0	6	(16|1) (19|1) (22|1)
16	52	73	215	23	6	4	0	2	(8|1) (21|1)
16	52	73	216	15	4	3	0	7	
16	52	73	217	23	7	2	0	2	(8|1) (9|1) (15|1)
17	53	74	218	5	5	5	0	3	(16|1) (21|1)
17	53	74	219	17	3	5	0	5	(8|1) (21|1)

17	53	74	220	6	3	4	0	2	(10|1) (15|1) (19|1)
17	53	74	221	17	6	6	0	6	(20|2) (21|1)
17	54	75	222	10	4	4	0	0	(18|1)
17	54	75	223	10	7	1	0	2	
17	54	75	224	8	5	4	0	0	(21|1)
17	55	76	225	5	4	1	0	1	(8|1)
17	55	76	226	4	6	4	0	3	(14|1) (21|1)
17	55	76	227	19	8	3	0	2	
17	55	76	228	19	3	1	0	5	(8|1) (19|1)
18	56	77	229	8	10	3	0	4	
18	56	77	230	14	5	1	0	7	(8|1)
18	56	77	231	4	4	4	0	0	(13|2) (14|1)
18	57	78	232	5	6	5	0	4	(13|1)
18	57	78	233	5	10	1	0	2	(16|1) (19|1)
18	57	79	234	7	4	6	0	4	(15|1) (21|1)
18	57	79	235	7	4	4	0	3	(13|1) (14|1) (19|1)
19	58	80	236	4	5	1	0	4	(16|1) (21|1)
19	58	80	237	5	5	4	0	5	(14|1) (20|2) (21|2)
19	58	80	238	4	3	2	0	6	(19|1)
19	58	80	239	2	5	2	0	4	(15|1)
19	59	81	240	3	6	3	0	2	(21|1)
19	59	81	241	8	5	5	0	5	(20|2)
19	59	82	242	10	10	3	0	3	
19	59	82	243	2	4	5	0	2	(13|1)
19	59	82	244	2	2	6	0	4	(20|1)
19	59	82	245	6	4	1	0	3	(12|1) (13|1) (16|1) (20|1)
20	60	83	246	3	3	4	0	3	(9|2) (16|1)
20	60	83	247	9	6	3	0	4	(8|1) (10|1) (21|1)
21	61	84	248	9	4	4	0	6	(9|1) (14|1)
21	61	84	249	3	8	4	0	4	(9|2) (14|1) (19|2)
21	61	84	250	2	1	2	0	5	(19|2) (20|1)

21	61	84	251	7	3	1	0	4	(13|1)
21	62	85	252	9	7	2	0	7	(15|1)
21	62	86	253	7	5	2	0	3	(15|1)
21	63	87	254	11	6	1	0	4	
21	63	87	255	2	3	2	0	5	(13|1)
21	63	87	256	5	4	2	0	3	(13|1)
21	63	87	257	3	4	2	0	4	(14|1) (19|1)
21	63	87	258	3	3	2	0	6	(10|1) (16|1) (21|1)
22	64	88	259	3	6	3	0	6	(9|1) (20|1)
22	64	88	260	3	4	3	0	3	(8|1)
22	64	88	261	8	3	3	0	4	(13|1) (16|1)
22	65	89	262	4	2	1	0	3	
22	65	89	263	3	5	1	0	4	
22	65	90	264	7	7	1	0	2	(14|2) (15|1) (16|1)
23	66	91	265	11	4	4	0	2	(19|1)
23	66	91	266	5	5	3	0	1	(9|1)
23	66	91	267	10	6	2	0	3	(14|1) (20|1)
23	67	92	268	6	6	2	0	0	(14|1)
23	67	92	269	4	3	2	0	4	(20|2)
23	67	92	270	3	7	3	0	2	(9|1) (14|1)
23	67	92	271	4	2	4	0	0	(10|1)
24	68	93	272	4	5	1	0	5	(9|1) (20|1)
24	68	93	273	3	3	3	0	4	(12|1) (16|1) (19|2)
24	68	93	274	3	2	2	0	5	(10|1) (12|1) (21|1)
24	68	93	275	3	5	3	0	6	
24	68	94	276	4	3	1	0	6	(10|1) (20|1)
24	68	94	277	6	8	1	0	5	(14|1)
24	69	95	278	4	6	1	0	4	(16|1) (21|1)
24	69	95	279	4	6	3	0	3	(15|1)
24	69	95	280	10	5	1	0	7	(16|1)
24	69	95	281	4	0	2	0	3	(12|1)
24	69	95	282	4	7	4	0	7	(16|1)

24	69	96	283	3	3	3	0	4	(9|1) (14|1) (20|1)
24	69	96	284	6	7	3	0	6	
24	69	96	285	12	3	3	0	7	
24	70	97	286	3	3	2	0	2	(10|1)
24	70	97	287	7	8	2	0	6	(12|1) (15|1) (20|1)
24	70	97	288	3	7	1	0	4	
24	70	97	289	3	6	3	0	5	
24	71	98	290	3	2	4	0	4	(14|1) (21|1)
24	71	98	291	5	4	2	0	1	(16|1)
24	72	99	292	4	6	3	0	6	(21|1)
24	72	99	293	5	4	4	0	3	(8|1)

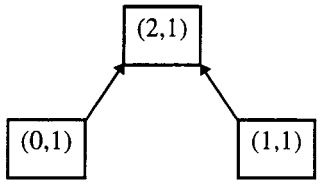
Table A.11. Routing Structure and Operation Related Data - Problem 3

Operation	Setup times (preceding operation | setup)
12	(24|6) (118|6) (140|8) (151|6) (225|6)
23	(34|6) (36|4) (69|7) (73|3) (88|5) (137|5) (150|5) (190|5) (194|6) (208|7) (222|4) (224|6) (231|6) (268|6) (271|5)
24	(118|6) (134|5) (140|5) (156|8) (225|7) (266|6)
34	(23|3) (36|4) (69|3) (73|4) (88|6) (137|1) (150|5) (190|6) (194|4) (208|4) (222|5) (224|5) (268|3) (271|6)
36	(69|8) (73|8) (88|4) (137|6) (150|1) (190|6) (194|8) (208|3) (224|3) (268|2) (271|3)
65	(12|4) (24|4) (118|3) (134|4) (140|4) (156|7) (225|4) (266|4)
69	(23|7) (34|6) (36|2) (73|0) (137|8) (208|5) (222|8) (224|5) (231|4) (271|4)
73	(23|5) (137|7) (190|4) (194|5) (222|6) (268|5) (271|9)
88	(23|9) (34|6) (36|4) (73|5) (137|6) (150|4) (194|7) (208|1) (222|6) (231|7) (268|4)
118	(24|7) (65|5) (134|7) (140|7) (151|7) (225|3) (266|5)
134	(12|3) (24|4) (65|5) (118|6) (140|4) (151|6) (225|4) (266|8)
137	(23|9) (34|3) (36|7) (73|5) (88|4) (150|4) (190|3) (194|7) (208|3) (222|9) (231|4) (268|3) (271|4)
140	(12|5) (24|4) (118|8) (134|4) (156|3)
150	(23|7) (34|3) (36|3) (69|2) (73|3) (88|9) (194|3) (208|4) (222|6) (224|10) (231|3) (268|8) (271|4)
151	(12|3) (65|6) (118|9) (140|5) (266|7)
156	(12|6) (24|6) (65|3) (134|6) (140|7) (151|8) (225|1) (266|1)
190	(34|3) (36|3) (73|4) (88|9) (137|6) (150|6) (194|8) (208|9) (222|4) (224|6) (231|8) (268|4)
194	(34|8) (36|3) (69|7) (73|6) (88|6) (137|3) (150|2) (190|6) (208|2) (222|7) (231|7) (268|10) (271|7)
208	(34|6) (36|4) (69|2) (73|4) (88|0) (137|3) (190|5) (194|5) (224|2) (231|5) (268|3) (271|7)
222	(23|7) (34|6) (69|6) (73|6) (88|3) (137|5) (190|6) (194|8) (208|5) (224|5) (231|7) (268|8) (271|5)
224	(23|2) (34|3) (69|4) (73|3) (88|6) (137|4) (150|6) (190|8) (222|8) (231|3) (268|7) (271|2)
225	(12|5) (24|8) (65|7) (118|5) (140|7) (151|3) (266|1)
231	(23|5) (36|8) (73|6) (88|8) (137|5) (150|5) (190|5) (194|2) (208|7) (268|5) (271|3)
266	(12|8) (24|6) (65|10) (118|4) (140|6) (151|8) (156|4) (225|9)
268	(23|0) (34|2) (36|3) (69|4) (73|8) (88|2) (137|7) (150|8) (190|9) (194|4) (208|4) (224|5) (231|3) (271|3)
271	(23|0) (34|3) (36|4) (73|4) (137|6) (150|4) (190|8) (194|3) (208|6) (224|6) (231|6)

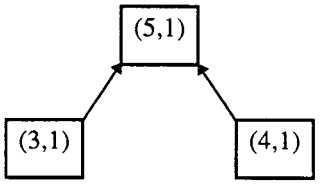
Table A.12. Operations with Sequence Dependent Setup Times - Problem 3

Problem 4: Problem 4 is defined as follows.

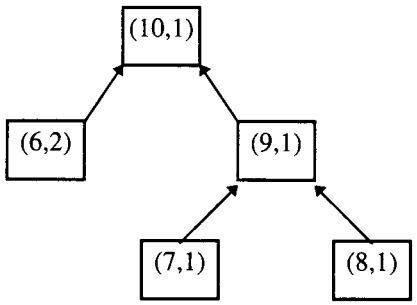
Job 0:



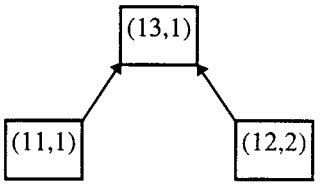
Job 1:



Job 2:



Job 3:



Job 4:

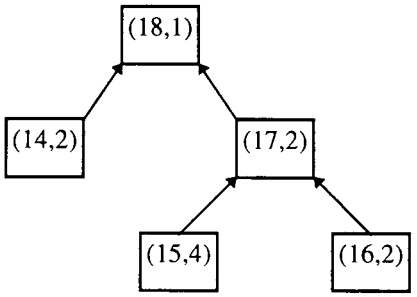


Figure A.4. BOMs - Jobs from Problem 4

Job	batch size	due date	ready time
0	10	1008	0
1	13	480	0
2	11	672	0
3	4	1728	80
4	12	1152	0

Table A.13. Job Related Dada - Problem 4

Resource class	Resource (type|unit)	Scheduled breaks (start time|duration)
Machines	(0|0)	(813|24)
	(1|0)	(83|25) (1356|37)
	(2|0)	(488|10) (1866|13)
	(2|1)	(491|27) (1838|33)
	(3|0)	(442|32) (1398|48)
	(3|1)	(56|23) (1351|54)
	(3|2)	(403|32) (1611|26)
	(4|0)	(55|33) (1276|29)
	(4|1)	(428|37) (1917|26)
	(4|2)	(327|25) (1456|36)
	(5|0)	(1221|30)
	(5|1)	(722|36) (2054|29)
Other resources	(6|0)	(1152|27)
	(7|0)	(118|42) (1336|35)
	(7|1)	(558|38) (1832|46)
	(7|2)	(513|44) (1817|36)
	(7|3)	(0|40) (1144|27)
	(8|0)	(371|32) (1531|14)
	(8|1)	(265|26) (1419|23)
	(8|2)	(682|30) (2032|18)
	(8|3)	(1016|33)
	(9|0)	(847|25)
	(9|1)	(839|28)
	(9|2)	(820|12)
	(9|3)	(699|15) (1902|29)
	(10|0)	(673|43) (1940|42)
	(10|1)	(428|37) (1749|43)
	(10|2)	(1012|26)
	(10|3)	(829|23)

(11|0)	(13|40) (1409|34)
(12|0)	(731|39) (1802|23)
(12|1)	(192|32) (1340|33)
(12|2)	(1140|35)
(12|3)	(148|18) (1450|26)
(13|0)	(688|25) (2105|29)
(13|1)	(191|18) (1661|32)
(13|2)	(1239|35)
(13|3)	(741|34) (2059|28)

Table A.14. Resource Related Data - Problem 4

Part	Sub-process	Route	Operation	Minimum transport batch	Setup time	Unit execution time	Operation ready time	Machine type required	Other resources required (type|quantity)
0	0	0	0	7	5	3	0	2	(10|1)
0	1	1	1	2	5	4	0	4	(9|1) (10|1) (12|1)
0	1	1	2	3	6	4	0	5	
0	2	2	3	8	5	3	0	2	
0	2	2	4	4	4	3	0	3	(8|1) (10|1)
0	2	2	5	4	4	3	0	1	(9|1) (12|1) (13|1)
0	3	3	6	5	5	1	0	5	(8|1) (10|1) (12|2)
0	3	3	7	3	4	6	0	4	
0	3	4	8	2	5	3	0	2	(7|1) (10|2) (13|1)
0	4	5	9	7	6	1	0	3	(8|1) (10|1) (13|1)
0	4	5	10	6	6	2	0	2	(8|1) (10|1)
1	5	6	11	7	3	1	0	5	(10|1)
1	5	6	12	2	3	2	0	3	(10|1) (13|1)
1	5	6	13	3	6	4	0	4	(7|1)
1	6	7	14	4	4	3	0	2	
1	6	7	15	3	2	3	0	5	(8|1)
1	7	8	16	3	5	3	0	4	(10|1)
1	7	8	17	2	6	3	0	5	(7|1)
1	8	9	18	8	5	1	0	2	(7|1)
1	8	9	19	2	5	3	0	3	(10|1)
1	8	10	20	5	4	4	0	2	(9|1) (10|2) (12|1)
2	9	11	21	2	5	2	0	5	(10|1) (13|1)
3	10	12	22	4	3	1	0	4	(7|1) (8|1) (9|1) (10|1)
3	10	13	23	4	4	1	0	3	(8|1)
3	10	13	24	6	5	5	0	2	(10|1) (12|1) (13|1)
3	10	13	25	8	4	1	0	3	(9|1) (10|1) (13|1)
3	11	14	26	8	5	1	0	5	(10|1) (12|1) (13|1)
3	11	14	27	3	7	4	0	3	(9|2) (12|1)

3	11	15	28	3	6	1	0	2	(8|1) (9|1)
3	11	15	29	11	5	1	0	4	(10|1) (12|1)
3	12	16	30	10	4	2	0	5	(7|2)
3	12	16	31	6	4	2	0	2	(9|1)
3	12	16	32	6	3	4	0	4	(7|1) (12|1)
3	12	17	33	6	4	1	0	5	
3	13	18	34	4	5	4	0	3	
3	13	18	35	3	5	1	0	1	(10|1)
4	14	19	36	6	6	3	0	3	(10|1)
4	14	19	37	12	5	1	0	4	(8|1) (10|1)
4	14	19	38	9	5	1	0	2	
4	14	20	39	5	4	6	0	2	(9|1) (10|1)
4	14	20	40	3	5	1	0	4	(9|1) (13|1)
4	14	20	41	5	5	3	0	2	(9|1) (12|1)
4	15	21	42	3	5	2	0	1	(12|1)
4	15	21	43	10	4	1	0	2	(7|1) (8|1) (10|1)
4	15	22	44	5	6	5	0	5	(10|1)
4	15	22	45	6	3	2	0	0	(8|1) (10|1) (13|2)
4	15	22	46	8	5	2	0	2	(8|1) (12|1)
4	16	23	47	3	7	1	0	3	(8|1) (13|1)
4	17	24	48	5	7	2	0	4	
4	17	25	49	8	3	1	0	5	(8|1)
4	17	25	50	5	3	2	0	4	(8|1) (13|1)
4	17	25	51	3	5	4	0	2	(6|1) (10|1) (12|1)
5	18	26	52	5	5	3	0	3	(8|2)
5	18	26	53	5	7	5	0	5	
5	19	27	54	3	6	1	0	3	
5	19	27	55	3	7	3	0	0	(6|1) (10|1)
6	20	28	56	14	5	2	0	4	(10|1)
6	20	28	57	5	5	4	0	2	(8|1) (13|1)
6	20	29	58	7	5	3	0	3	(8|1) (10|1)
6	20	29	59	7	6	1	0	4	(6|1) (8|1) (10|1)

7	21	30	60	10	4	1	0	3	(10|1)
7	21	30	61	2	6	3	0	5	(8|1) (10|1) (12|1) (13|2)
7	21	31	62	4	4	1	96	2	(7|1)
7	21	31	63	4	5	4	0	0	(10|1) (12|1)
7	21	31	64	3	6	3	0	2	(8|1) (10|1)
7	22	32	65	2	6	3	0	1	(8|1) (9|1) (10|1) (12|1)
7	22	32	66	4	5	1	0	5	(13|1)
8	23	33	67	6	4	2	0	1	
8	24	34	68	2	6	4	0	2	(8|1) (10|1) (13|1)
8	24	34	69	7	4	5	0	0	(12|1) (13|1)
8	24	34	70	5	5	1	0	2	(8|1)
8	24	35	71	2	4	3	0	5	(8|1)
8	24	35	72	2	5	3	0	4	(9|1) (13|1)
9	25	36	73	6	7	2	0	0	(7|1) (10|2)
9	25	36	74	2	4	2	0	5	
9	26	37	75	4	6	4	0	2	(8|1) (10|1) (12|1)
9	26	37	76	5	6	5	0	4	(9|1) (10|1)
9	27	38	77	9	5	1	0	1	(8|1) (12|1)
9	28	39	78	3	4	4	0	4	
9	28	39	79	2	6	4	0	3	(8|1)
9	28	39	80	3	3	4	0	4	(9|1)
9	28	40	81	2	5	1	0	5	(10|1) (12|1) (13|1)
9	28	40	82	4	6	2	0	3	
9	28	40	83	2	6	3	0	1	(7|2)
10	29	41	84	7	4	1	0	5	(8|1)
10	29	41	85	7	6	4	0	4	(13|1)
10	29	41	86	3	6	3	0	3	(8|1) (10|1) (13|1)
10	30	42	87	3	7	3	0	5	(12|1)
10	30	42	88	3	5	3	0	3	(8|1) (10|1)
10	30	42	89	11	4	4	0	5	(13|1)

10	31	43	90	7	6	3	0	3	(9I1) (10I1)
10	31	43	91	3	6	1	0	0	(7I1) (9I1)
10	31	44	92	7	4	3	0	4	(7I1) (8I1)
10	31	44	93	7	7	1	0	5	
10	32	45	94	3	5	1	0	3	(9I1) (10I1)
10	32	45	95	2	5	3	0	5	(13I1)
10	32	45	96	3	5	4	0	3	(8I1) (10I1)
11	33	46	97	1	5	2	0	5	(7I2) (8I1)
11	33	46	98	1	6	2	0	4	(12I1) (13I1)
11	34	47	99	1	5	3	0	5	(10I1) (13I1)
11	34	47	100	1	5	3	0	4	(8I1) (12I1)
12	35	48	101	3	5	4	0	5	(6I1)
12	36	49	102	2	5	2	0	2	(8I1) (10I1)
12	36	49	103	4	5	5	0	3	(12I1)
13	37	50	104	2	6	4	0	2	(8I1) (9I1) (10I1)
13	37	50	105	1	4	1	0	4	(12I1)
13	37	50	106	1	5	3	0	5	
13	38	51	107	1	5	2	0	3	(7I1) (8I1) (9I2) (10I1)
13	38	52	108	3	6	1	0	3	(9I1) (12I1) (13I1)
13	39	53	109	1	6	1	0	5	
13	39	53	110	3	4	3	0	4	(8I1) (10I2)
13	39	53	111	1	5	2	0	0	(8I1) (9I1)
13	40	54	112	1	3	3	0	3	(8I1) (10I1)
13	40	54	113	3	5	4	0	4	(10I2) (12I1)
13	40	54	114	1	5	3	0	5	(8I1) (13I1)
14	41	55	115	10	5	1	0	3	(7I1) (8I1) (10I1)
14	41	55	116	18	5	4	0	2	(7I1) (10I1)
14	41	56	117	5	5	2	0	3	(8I1) (10I1) (12I1)
14	41	56	118	19	4	1	55	0	(8I1) (10I1) (13I1)
14	42	57	119	6	5	1	0	1	(8I1)
14	42	57	120	7	4	3	0	4	(10I1)

14	43	58	121	5	6	1	0	3	(8|1) (9|1) (10|1) (12|1)
14	44	59	122	9	4	3	0	5	(8|1) (10|1)
14	44	59	123	6	5	1	0	2	(10|1)
14	44	59	124	8	4	1	0	4	(12|2) (13|1)
15	45	60	125	26	5	3	0	4	(7|1) (8|1) (10|1) (13|1)
15	45	60	126	29	5	3	0	5	(10|1) (12|2) (13|1)
15	45	60	127	35	6	2	0	4	(8|1) (9|1) (10|1)
15	45	61	128	18	4	3	0	4	(13|1)
15	45	61	129	28	5	3	0	1	(8|1) (9|1)
15	45	61	130	11	7	2	0	2	(12|1) (13|1)
15	46	62	131	11	5	1	0	5	(10|1)
15	46	62	132	14	5	2	0	2	(7|1) (9|2) (10|1)
15	46	63	133	14	3	4	0	5	(7|1) (8|1) (13|1)
15	46	63	134	19	5	1	0	4	(8|1) (10|1) (13|1)
15	47	64	135	26	5	1	0	3	(13|1)
15	47	64	136	31	6	2	0	2	(8|1) (12|1)
16	48	65	137	6	5	1	0	4	(9|1) (10|1) (12|1)
16	48	65	138	6	7	3	0	3	(9|1)
16	49	66	139	5	5	1	0	4	(8|1) (9|1) (12|1) (13|1)
16	49	66	140	10	6	4	0	2	(9|1)
16	50	67	141	13	7	1	0	5	(8|1) (10|1) (12|1) (13|1)
16	50	67	142	7	6	2	0	4	(12|1)
16	51	68	143	6	3	3	0	5	
16	51	68	144	5	5	1	0	2	(10|1)
17	52	69	145	5	4	1	28	3	(9|1) (10|1)
17	52	69	146	13	5	4	0	5	(12|1)
17	52	69	147	11	5	1	0	4	(10|1) (13|1)
17	52	70	148	17	5	2	0	3	(8|2) (9|1) (12|1)

17	52	70	149	5	4	1	0	5	(8|1) (10|1)
17	53	71	150	6	4	4	0	3	(8|1) (12|1)
17	53	71	151	6	5	1	0	4	(7|1)
17	53	71	152	8	5	4	0	1	(8|1) (10|1)
17	53	72	153	5	3	5	0	3	(8|2) (10|1) (12|1)
17	54	73	154	12	6	4	0	0	(12|2)
17	54	73	155	20	6	3	0	2	(8|1)
17	54	73	156	9	5	3	0	3	(8|1) (10|1)
17	54	74	157	14	3	1	0	4	(10|1) (12|1) (13|1)
17	54	74	158	5	4	1	0	0	(9|1) (10|1)
17	55	75	159	6	6	3	0	4	(9|1) (10|1)
17	55	75	160	11	5	1	0	3	(8|1) (12|1) (13|1)
17	55	75	161	8	4	3	0	5	(9|1) (10|1)
17	56	76	162	5	4	2	0	3	
17	56	76	163	14	5	3	0	4	(9|2)
17	56	77	164	14	5	2	0	2	(8|2) (9|1) (10|2) (12|1)
17	56	77	165	21	6	4	0	3	(10|1)
17	56	77	166	5	6	6	0	5	(12|1) (13|1)
18	57	78	167	6	6	4	0	3	(9|1) (10|1) (13|1)
18	58	79	168	4	6	4	0	5	(8|1) (10|1) (12|1)
18	58	79	169	3	5	2	0	3	(7|1)
18	58	80	170	3	6	3	0	4	(9|1) (10|1) (12|2)
18	58	80	171	5	5	1	0	3	
18	58	80	172	5	3	1	0	5	(7|1) (8|1) (10|1) (12|1)
18	59	81	173	3	5	2	0	2	(10|1)
18	60	82	174	6	6	3	0	3	(8|1)

Table A.15. Routing Structure and Operation Related Data - Problem 4

Operation	Setup times (preceding operation | setup)
5	(42|5) (65|5) (67|5) (77|6) (83|5) (119|5)
35	(5|4) (42|6) (65|3) (67|4) (77|3) (83|4) (119|6) (129|3) (152|4)
42	(5|4) (35|4) (67|6) (77|6) (83|4) (119|4) (129|4)
45	(55|4) (69|7) (111|5) (154|5)
55	(45|5) (69|5) (111|5)
65	(5|7) (35|5) (42|5) (67|5) (77|4) (83|5) (119|7) (152|4)
67	(5|5) (35|6) (42|7) (65|6) (77|7) (83|5) (119|7) (152|5)
69	(45|5) (55|5) (111|5) (154|6)
77	(5|3) (35|3) (42|6) (65|4) (67|6) (83|6) (119|6) (129|6) (152|3)
83	(35|5) (42|7) (65|5) (67|4) (77|4) (119|4) (152|5)
111	(45|3) (55|6) (69|6) (154|4)
119	(5|7) (35|4) (42|3) (65|7) (67|7) (83|6) (129|4) (152|4)
129	(5|4) (35|6) (42|7) (67|7) (77|6) (83|4) (119|3) (152|7)
152	(5|4) (35|6) (42|4) (65|6) (67|4) (77|6) (83|7) (129|7)
154	(45|7) (55|4) (111|4)

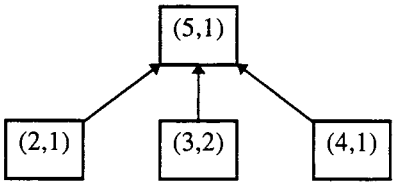
Table A.16. Operations with Sequence Dependent Setup Times - Problem 4

Problem 5: Figure A.5 and tables A.17 - A.20 describe problem 5.

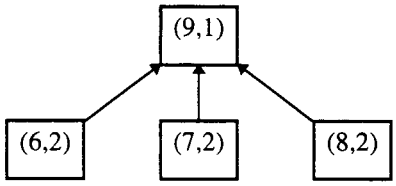
Job 0:

Job 1 :

Job 2:



Job 3:



Job 4:

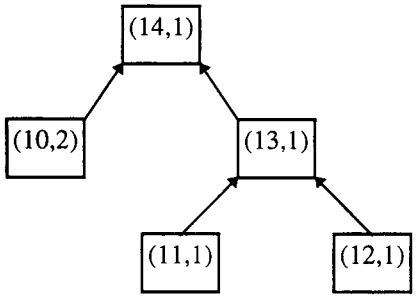


Figure A.5. BOMs - Jobs from Problem 5

Job	batch size	due date	ready time
0	13	912	0
1	6	1536	0
2	6	1968	0
3	10	1616	85
4	14	512	0

Table A.17. Job Related Dada - Problem 5

Resource class	Resource (type|unit)	Scheduled breaks (start time|duration)
Machines	(0|0)	(304|47) (1551|7)
	(1|0)	(737|42)
	(2|0)	(519|26) (1649|38)
	(2|1)	(629|54)
	(2|2)	(929|36)
	(3|0)	(1088|30)
	(3|1)	(636|42)
	(4|0)	(26|41) (1351|43)
	(4|1)	(265|44) (1557|32)
	(4|2)	(552|38) (1694|47)
	(4|3)	(104|37) (1233|45)
	(5|0)	(1047|27)
	(5|1)	(1458|22)
Other resources	(6|0)	(92|21) (1325|37)
	(6|1)	(1164|33)
	(7|0)	(1084|43)
	(7|1)	(966|26)
	(8|0)	(53|27) (1400|31)
	(8|1)	(99|36) (1263|26)
	(8|2)	(793|13)
	(8|3)	(59|27) (1394|38)
	(9|0)	(882|21)
	(9|1)	(186|25) (1483|43)
	(9|2)	(1166|27)
	(9|3)	(26|30) (1196|21)
	(10|0)	(606|38)
	(10|1)	(996|36)
	(10|2)	(554|30) (1604|26)
	(10|3)	(680|32)
	(11|0)	(575|35) (1618|32)
	(11|1)	(887|10)

	(11|2)	(408|41) (1709|33)
	(11|3)	(768|41)
	(12|0)	(27|25) (1084|40)
	(12|1)	(594|10)
	(13|0)	(557|36) (1805|28)
	(13|1)	(43|27) (1234|16)
	(13|2)	(1070|19)
	(13|3)	(752|33)

Table A.18. Resource Related Data - Problem 5

Part	Sub-process	Route	Operation	Minimum transport batch	Setup time	Unit execution time	Operation ready time	Machine type required	Other resources required (type|quantity)
0	0	0	0	5	4	1	0	5	
0	0	0	1	4	5	5	0	2	
0	0	0	2	9	6	3	0	4	
0	1	1	3	8	3	5	0	3	(11|1)
0	1	1	4	9	7	3	0	0	
0	1	2	5	3	6	1	0	5	
0	1	2	6	5	5	4	0	2	
1	2	3	7	3	4	1	0	5	(10|1)
1	2	3	8	4	6	3	0	2	
1	3	4	9	3	7	1	0	5	
1	3	4	10	1	4	3	0	1	
1	3	5	11	3	3	4	0	4	(11|1)
1	3	5	12	1	7	4	64	5	
1	3	5	13	3	4	2	0	4	
1	4	6	14	4	2	1	0	5	(10|1)
1	4	7	15	2	5	2	0	5	
1	5	8	16	2	6	1	0	1	
1	5	8	17	1	2	2	0	3	
2	6	9	18	2	5	1	0	3	(10|1)
2	6	9	19	2	10	2	0	1	(9|1)
2	7	10	20	2	4	3	0	3	
2	7	10	21	5	2	3	0	5	(9|1) (10|1)
2	7	11	22	2	3	3	0	4	
3	8	12	23	5	4	2	0	4	
3	8	12	24	4	3	2	0	5	(9|1)
3	9	13	25	6	7	4	0	4	(9|1)
3	9	13	26	8	9	3	0	3	
3	9	13	27	3	5	1	0	4	
3	10	14	28	12	6	4	0	5	(12|1)

3	10	14	29	2	4	3	0	3	
3	10	14	30	3	0	4	0	2	(11|1)
4	11	15	31	2	7	2	0	4	
4	11	15	32	3	7	2	0	5	
4	12	16	33	4	8	4	0	2	
5	13	17	34	5	4	4	0	3	
5	14	18	35	1	7	2	0	1	(9|1)
5	15	19	36	2	3	3	0	4	
5	15	19	37	2	4	3	68	3	(9|1)
5	15	19	38	2	5	4	0	5	
5	15	20	39	2	6	4	0	4	
5	15	20	40	2	6	1	0	3	(10|1) (11|1)
5	16	21	41	4	5	3	0	1	
5	16	21	42	2	7	1	0	2	(9|1)
5	16	21	43	2	4	4	0	3	(13|1)
5	16	22	44	3	3	1	0	2	
5	16	22	45	2	7	3	0	3	(10|1)
5	16	22	46	3	2	2	0	4	(8|1) (12|1)
6	17	23	47	14	5	6	0	3	(9|1)
6	17	23	48	7	9	3	0	5	
6	17	23	49	4	6	3	0	0	
6	18	24	50	4	7	5	0	5	(9|1)
6	18	24	51	8	6	1	0	3	(11|1)
6	18	25	52	5	5	4	0	4	
6	18	25	53	8	7	5	0	5	(9|1)
6	18	25	54	4	5	3	33	1	(9|1)
6	19	26	55	5	4	1	0	5	
6	19	26	56	10	3	2	0	3	(7|1)
6	19	26	57	5	3	6	0	4	
6	20	27	58	4	3	3	0	5	(13|1)
6	21	28	59	6	1	3	0	4	
7	22	29	60	7	7	4	0	5	(6|1)

7	22	29	61	5	2	1	0	4	
7	22	29	62	9	9	4	0	5	(11|1)
7	23	30	63	19	5	6	0	1	
7	23	30	64	10	4	2	0	2	
7	23	30	65	6	4	1	0	4	
7	24	31	66	10	5	1	0	5	(9|1)
7	25	32	67	8	4	4	0	2	
7	25	32	68	4	7	2	0	0	
7	25	32	69	5	2	1	0	5	(11|1) (12|1)
8	26	33	70	5	9	3	0	3	
8	26	33	71	10	0	1	0	5	
8	26	34	72	6	7	4	0	5	
8	27	35	73	13	4	4	33	4	
8	27	35	74	12	3	3	0	2	
8	28	36	75	5	8	2	0	1	(8|1)
8	28	36	76	4	8	1	0	4	
9	29	37	77	4	0	4	0	3	
9	30	38	78	4	4	1	0	2	
9	30	38	79	4	6	1	0	3	
10	31	39	80	6	4	2	0	3	(10|1) (12|1)
10	31	39	81	23	3	1	0	2	(9|1)
10	32	40	82	9	6	2	0	3	
10	32	40	83	8	1	1	0	2	
10	32	41	84	13	1	4	0	4	(8|1)
10	32	41	85	9	4	1	87	5	
10	32	41	86	17	6	4	0	3	
10	33	42	87	9	4	2	0	4	(9|1)
10	33	42	88	18	4	3	0	3	(8|1)
11	34	43	89	3	4	4	0	5	
11	34	43	90	6	5	3	0	2	
11	34	44	91	5	4	1	0	0	
11	34	44	92	4	8	3	0	5	(13|1)

11	34	44	93	4	6	3	0	3	
11	35	45	94	5	7	3	0	4	
11	35	45	95	7	5	4	0	5	(12|1)
11	35	45	96	13	6	4	0	2	
11	35	46	97	5	7	2	0	4	
11	35	46	98	11	3	3	0	2	(10|1)
11	35	46	99	9	5	3	0	4	
11	36	47	100	4	0	2	0	1	
11	36	47	101	4	5	2	0	4	(9|1)
11	36	48	102	7	4	2	0	5	(9|1)
11	37	49	103	11	8	5	0	3	
11	37	50	104	6	4	2	0	1	(13|1)
11	37	50	105	6	6	2	0	4	(7|1)
11	38	51	106	5	5	6	0	2	
11	38	51	107	6	5	1	0	4	(13|1)
11	38	51	108	13	6	5	0	1	
12	39	52	109	6	2	4	0	2	
12	39	52	110	4	5	2	0	4	
12	39	53	111	9	3	5	0	4	(13|1)
12	39	53	112	7	8	2	0	2	
12	39	53	113	4	5	3	0	3	
12	40	54	114	3	1	3	0	5	
12	41	55	115	5	8	4	0	2	(7|1) (9|1) (12|1)
12	41	55	116	3	2	3	0	5	(8|1) (12|1)
13	42	56	117	7	6	3	0	0	(10|1)
13	42	56	118	3	5	3	0	2	
13	42	56	119	6	5	4	0	5	
13	42	57	120	3	2	5	0	5	
13	43	58	121	6	8	5	0	3	
14	44	59	122	5	7	3	0	4	
14	44	59	123	11	4	5	0	2	
14	45	60	124	4	5	1	0	0	

14	45	60	125	4	6	5	0	1	
14	46	61	126	12	1	1	0	2	
14	46	61	127	6	5	1	0	4	(13|1)
14	46	62	128	4	6	4	0	4	
14	46	62	129	3	6	1	0	5	
14	46	62	130	9	8	3	0	2	
14	47	63	131	3	4	5	0	3	(9|1) (11|1)
14	47	63	132	9	7	4	0	4	
14	47	63	133	4	2	3	0	1	

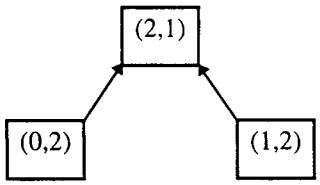
Table A.19. Routing Structure and Operation Related Data - Problem 5

Operation	Setup times (preceding operation | setup)
4	(49|3) (68|8) (91|8)
16	(41|0) (63|7) (75|4) (100|8) (108|7) (125|4) (133|4)
19	(16|4) (41|9) (63|7) (75|0) (100|6) (108|6) (125|7)
41	(16|8) (19|6) (63|6) (75|8) (100|4) (108|7) (125|0) (133|6)
49	(68|7) (91|0) (117|0) (124|4)
63	(41|6) (75|6) (100|2) (108|7) (125|4) (133|7)
68	(4|8) (49|4) (91|3) (117|0) (124|6)
75	(16|3) (19|7) (41|4) (63|6) (100|6) (108|6) (125|5)
91	(4|5) (49|3) (117|7)
100	(19|6) (63|7) (75|6) (108|3) (125|4) (133|7)
108	(16|7) (19|4) (63|4) (100|7) (125|3) (133|2)
117	(4|2) (68|2) (91|7) (124|3)
124	(49|6) (68|6) (117|8)
125	(16|5) (19|7) (41|7) (63|7) (75|7) (100|4) (133|1)
133	(16|6) (19|5) (41|4) (63|5) (75|8) (108|6) (125|3)

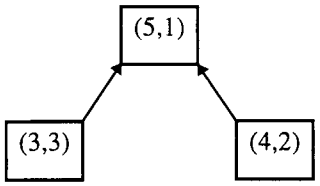
Table A.20. Operations with Sequence Dependent Setup Times - Problem 6

Problem 6: Finally problem 6 is described next.

Job 0:

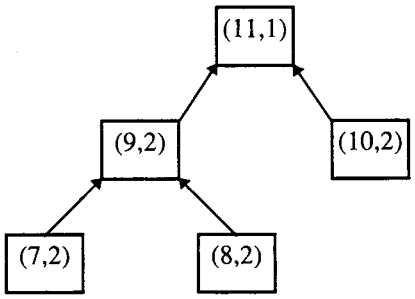


Job 1:



Job 2:

Job 3:



Job 4:

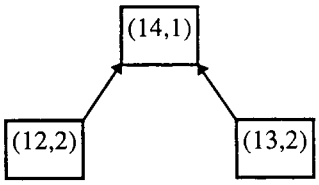


Figure A.6. BOMs - Jobs from Problem 6

Job	batch size	due date	ready time
0	10	992	0
1	9	1808	0
2	10	720	0
3	10	800	0
4	12	944	0

Table A.21. Job Related Dada - Problem 6

Resource class	Resource (type|unit)	Scheduled breaks (start time|duration)
Machines	(0|0)	(746|34) (2505|30)
	(1|0)	(1297|30)
	(2|0)	(821|31) (2442|13)
	(3|0)	(512|30) (1982|20)
	(4|0)	(726|38) (2279|19)
	(4|1)	(383|29) (2092|36)
	(5|0)	(1182|34)
	(5|1)	(549|21) (2040|45)
	(6|0)	(900|31) (2566|31)
	(7|0)	(906|26)
	(7|1)	(570|31) (1891|32)
	(8|0)	(1139|17)
	(8|1)	(143|23) (1801|23)
	(8|2)	(0|16) (1441|24)
Other resources	(9|0)	(557|24) (2156|31)
	(9|1)	(287|35) (1927|54)
	(9|2)	(1159|26)
	(9|3)	(1046|32)
	(10|0)	(92|17) (1429|36)
	(10|1)	(588|28) (2221|47)
	(10|2)	(1351|47)
	(10|3)	(325|24) (1579|27)
	(11|0)	(193|32) (1815|34)
	(11|1)	(941|1)
	(12|0)	(995|15)
	(12|1)	(17|48) (1445|24)
	(12|2)	(651|45) (2076|33)
	(12|3)	(731|36) (2357|33)
	(13|0)	(191|28) (1644|30)
	(13|1)	(658|34) (2312|26)
	(13|2)	(498|28) (2011|39)

(13|3)	(168|22) (1825|32)
(14|0)	(1542|45)
(14|1)	(518|33) (2411|41)
(15|0)	(973|33)
(15|1)	(977|7)
(15|2)	(192|26) (1643|19)
(15|3)	(1051|35)
(16|0)	(499|19) (2003|24)
(16|1)	(546|36) (2307|35)
(16|2)	(1221|12)
(16|3)	(540|23) (1913|21)
(17|0)	(798|37) (2185|40)
(17|1)	(507|34) (2191|30)
(17|2)	(0|40) (1855|7)
(17|3)	(92|21) (1613|25)
(18|0)	(347|32) (1579|42)
(18|1)	(551|30) (2126|20)

Table A.22. Resource Related Data - Problem 6

Part	Sub-process	Route	Operation	Minimum transport batch	Setup time	Unit execution time	Operation ready time	Machine type required	Other resources required (typelquantity)
0	0	0	0	5	7	4	0	3	(10l2)
0	0	0	1	7	1	1	0	6	(12l1)
0	0	0	2	7	4	1	65	2	(9l1) (17l2)
0	0	1	3	10	7	2	0	5	(16l1)
0	0	1	4	10	5	4	0	7	(10l1)
0	0	2	5	5	5	2	0	5	(9l1) (16l1) (17l1)
0	0	2	6	10	5	1	0	2	(9l1) (10l1) (16l1)
0	0	2	7	4	7	5	0	3	(9l2)
0	0	2	8	9	6	5	0	5	(9l1) (12l1)
0	0	2	9	4	4	2	0	7	
0	1	3	10	5	4	1	0	4	(10l1)
0	1	3	11	7	5	3	0	6	
0	1	3	12	6	5	1	0	3	(10l1) (12l1)
0	1	3	13	14	4	3	0	6	(10l1)
0	1	3	14	12	4	3	0	7	(10l1) (12l1)
0	2	4	15	7	6	3	0	5	(15l1)
0	2	4	16	17	7	2	0	6	
0	2	4	17	7	4	3	0	4	(12l2)
0	2	4	18	5	4	2	0	7	(15l1)
0	2	5	19	7	6	5	0	8	(10l1)
0	2	5	20	4	9	1	0	4	(10l1) (16l1)
0	2	5	21	8	4	2	0	6	(9l1) (10l1) (15l1) (16l1) (17l1)
0	2	6	22	7	5	1	0	8	(15l1)
0	2	6	23	5	4	4	0	3	
0	2	6	24	4	5	6	0	4	(12l1) (13l1)
0	2	6	25	6	4	2	0	5	(9l1) (10l1) (15l1)
0	2	6	26	5	2	1	0	8	(17l1)
0	2	7	27	11	5	4	0	3	(17l1) (18l1)

0	2	7	28	5	3	3	0	2	(15|1)
0	2	7	29	17	5	2	0	3	(12|1)
0	2	7	30	4	3	1	0	5	(9|1) (10|1)
0	3	8	31	10	5	1	0	0	(9|1)
0	3	8	32	10	3	4	0	7	(9|1) (10|1)
0	4	9	33	8	6	1	0	3	(9|1)
0	4	9	34	4	7	4	0	5	(10|1)
0	4	9	35	7	7	4	0	4	(15|1)
0	4	9	36	6	5	2	0	2	(15|1)
0	4	10	37	5	2	5	0	8	
0	4	10	38	7	4	2	0	3	
0	4	10	39	7	0	4	0	5	(9|1) (16|1)
0	4	10	40	5	9	2	0	6	(10|1) (17|1)
0	4	10	41	5	4	2	0	2	(10|1) (16|1) (17|2)
0	4	11	42	6	3	3	0	0	(10|1) (13|1)
0	4	11	43	12	4	5	0	8	(9|1)
0	4	11	44	4	8	5	0	7	(10|1)
1	5	12	45	6	6	2	0	6	
1	5	12	46	6	6	2	0	8	
1	5	12	47	4	3	1	0	7	(9|1) (14|1) (17|1)
1	5	12	48	12	5	2	0	4	(15|1)
1	5	12	49	6	6	1	0	3	(17|1)
1	5	13	50	5	6	4	0	4	(10|1) (16|1)
1	5	13	51	5	5	2	0	6	(15|1) (17|1)
1	5	13	52	12	4	3	0	7	(17|1)
1	5	14	53	6	5	4	0	8	(9|1) (10|1)
1	5	14	54	5	6	2	0	5	
1	5	14	55	10	7	4	0	2	(9|2) (10|1) (17|1)
1	5	14	56	7	5	5	0	4	
1	5	15	57	18	6	3	0	0	(15|1)
1	5	15	58	5	5	1	0	3	
1	5	15	59	6	7	4	0	4	(10|1) (12|1)

1	5	15	60	8	7	3	0	1	
1	6	16	61	7	3	3	0	8	
1	6	16	62	9	2	4	0	4	(12|2)
2	7	17	63	8	6	1	0	3	(10|1) (15|1) (17|1)
2	7	17	64	2	10	4	0	0	(16|1)
2	7	17	65	2	3	1	0	6	(10|1) (13|1)
2	8	18	66	3	4	3	0	5	(9|1)
2	8	18	67	3	2	4	0	6	(17|1)
2	8	18	68	4	8	1	0	8	
2	9	19	69	6	4	3	0	3	(9|1) (16|1)
2	9	19	70	5	9	1	0	6	(16|2) (18|1)
2	9	19	71	3	4	2	0	5	
2	9	19	72	2	2	3	0	4	(12|1) (13|2) (15|1)
2	9	20	73	6	7	3	0	5	(9|1)
2	9	20	74	3	5	2	0	0	(9|1) (16|1)
2	9	20	75	4	6	3	0	1	(9|1) (10|1) (15|1)
2	9	20	76	2	2	1	0	4	(9|1)
2	9	20	77	9	5	3	0	8	(10|1) (13|1) (15|2)
2	9	21	78	2	2	1	0	6	(10|1) (12|1)
2	9	21	79	3	2	3	0	3	(17|1)
2	9	21	80	2	6	1	0	8	(9|1) (15|1)
2	10	22	81	3	7	3	0	2	(16|1)
2	10	22	82	2	5	4	0	6	(9|1) (13|2) (16|1)
2	10	23	83	3	4	3	0	5	(10|2) (11|1) (17|1)
2	10	23	84	5	4	2	0	7	(11|1) (16|1)
3	11	24	85	6	7	3	0	7	
3	11	24	86	9	5	4	0	8	(9|1) (10|1) (12|2) (16|1)
3	11	25	87	6	5	4	0	6	
3	11	25	88	5	4	4	0	0	(10|1)
3	11	25	89	11	1	4	0	2	(10|1)
3	12	26	90	10	2	5	0	7	(10|1)

3	12	26	91	9	7	1	0	4	(9|1)
3	12	27	92	11	5	4	0	3	(10|1)
3	12	27	93	26	7	4	0	7	(12|1)
3	12	27	94	8	7	3	0	4	(15|1)
3	12	27	95	6	6	1	0	6	
3	12	28	96	7	4	4	0	6	(9|1) (12|1)
3	12	28	97	20	5	3	0	4	(16|1)
3	12	28	98	7	5	4	0	8	(9|1)
3	12	28	99	15	8	1	0	6	
3	12	28	100	6	9	4	0	1	(9|1)
3	13	29	101	12	6	4	0	8	(15|1) (16|1)
3	13	29	102	27	7	4	0	6	(9|1) (17|1)
3	13	29	103	24	5	1	0	5	(9|1) (18|1)
3	13	30	104	8	6	4	0	7	(10|1) (15|1) (18|1)
3	13	30	105	18	5	1	0	1	
3	13	30	106	6	6	3	0	2	(17|1)
3	13	31	107	6	6	2	0	5	
4	14	32	108	4	6	1	0	1	(16|2) (17|1)
4	14	32	109	6	7	2	0	3	
4	14	32	110	5	7	1	0	6	
4	14	33	111	5	5	1	0	1	(9|1)
4	14	33	112	8	6	4	0	7	(10|1)
4	15	34	113	10	8	3	0	0	
4	15	34	114	4	6	5	0	2	(9|1) (10|1) (13|1)
4	15	34	115	4	6	3	0	8	(10|1)
4	15	34	116	4	3	3	0	2	
4	15	35	117	15	6	2	0	1	(9|1) (17|1)
4	15	35	118	7	8	1	0	7	(18|1)
4	15	35	119	4	6	3	0	6	(10|1) (16|2)
5	16	36	120	9	6	4	0	8	(9|1) (14|1)
5	16	37	121	2	7	1	0	7	(15|1)
5	16	38	122	3	5	3	0	0	(10|1)

5	16	38	123	4	5	3	0	8	(9I1)
5	17	39	124	2	6	1	0	5	(10I1)
5	17	39	125	4	5	1	0	3	(15I1)
5	17	39	126	3	3	3	0	8	(10I1)
5	17	39	127	3	7	4	0	5	(9I1) (10I1) (11I2) (15I1)
5	17	40	128	2	5	2	0	5	(9I1) (17I1)
5	17	40	129	2	0	3	0	8	
5	17	40	130	2	7	3	0	4	
5	18	41	131	3	8	2	0	6	(13I1)
5	18	41	132	2	4	4	0	8	(9I1)
5	18	41	133	5	6	1	0	5	(9I2)
5	18	41	134	7	4	2	0	2	(10I1)
5	19	42	135	5	6	1	0	4	(9I1)
5	19	42	136	4	6	3	0	8	(9I1) (10I1)
5	19	42	137	6	5	6	0	0	(9I1)
5	19	42	138	6	8	3	0	2	
5	19	43	139	3	3	1	0	8	
5	19	43	140	2	2	3	0	3	(10I1)
5	20	44	141	3	3	2	0	4	(9I1) (10I1)
5	20	44	142	8	3	3	0	8	(9I1) (10I1)
5	20	44	143	2	2	6	0	5	(16I1)
5	20	44	144	2	5	1	0	8	
5	20	44	145	3	3	3	0	4	(10I1)
5	20	45	146	7	8	4	0	7	(9I1) (17I1)
5	20	45	147	9	6	1	0	2	(9I1) (10I1) (15I1)
5	20	45	148	2	5	3	0	1	(10I1)
5	20	45	149	3	3	2	0	6	
5	20	45	150	3	7	1	0	8	(9I1) (10I1) (16I1)
6	21	46	151	2	9	3	0	7	(14I1) (15I1) (16I1)
6	21	47	152	6	8	3	0	4	(17I1)
6	21	47	153	5	5	3	0	0	

6	21	47	154	3	8	4	0	8	(9|1)
6	21	48	155	3	3	4	0	6	(12|1)
6	21	48	156	3	4	4	0	7	(9|1) (10|1) (12|1)
6	22	49	157	5	6	3	0	2	(9|1)
6	22	49	158	2	4	3	0	5	
6	22	49	159	3	1	3	0	6	(11|2) (17|1)
6	22	49	160	7	5	1	0	7	
6	22	50	161	5	7	1	0	3	(9|1) (10|1)
6	22	50	162	2	6	3	0	4	(9|1) (10|1)
6	22	50	163	10	6	1	76	5	(9|1) (10|1)
6	22	50	164	3	2	5	0	7	
6	22	50	165	5	5	3	0	8	
7	23	51	166	5	4	4	0	2	(9|1) (16|2)
7	23	51	167	5	9	2	0	0	
7	23	51	168	10	9	3	0	7	(9|1)
7	23	52	169	8	3	2	0	7	
7	23	52	170	6	6	1	0	0	(9|1) (15|1)
7	23	52	171	16	6	4	0	4	(13|1)
7	23	52	172	6	3	1	0	6	(16|1)
7	23	52	173	4	5	2	0	8	(12|1)
8	24	53	174	5	0	4	0	6	
8	24	53	175	4	6	4	0	8	
8	24	53	176	7	5	5	0	4	(13|1)
8	25	54	177	5	6	4	0	5	(10|1) (15|1)
8	25	55	178	6	0	1	0	6	(10|1)
8	25	56	179	18	5	2	0	6	(10|1) (13|1)
8	26	57	180	8	2	2	0	3	(10|1) (15|1)
8	26	57	181	17	6	2	0	1	(10|1) (12|1)
8	26	57	182	8	3	3	0	4	(10|1) (17|1)
8	26	57	183	11	1	4	0	7	(9|2) (10|1) (17|1)
9	27	58	184	8	6	4	0	4	
9	27	59	185	10	3	1	0	4	(14|1) (18|1)

9	27	59	186	14	4	2	0	5	
9	27	59	187	7	4	1	0	3	(10|1)
9	27	60	188	9	4	3	0	4	
9	27	61	189	8	6	3	0	1	
9	27	61	190	16	9	3	0	2	
9	27	61	191	15	5	3	0	8	(10|1)
9	28	62	192	6	7	4	0	7	(9|1) (10|1)
9	28	62	193	4	6	1	0	8	(16|1)
9	28	62	194	19	4	2	0	3	(9|1) (10|1) (16|1) (17|1)
9	28	62	195	5	5	1	0	4	(17|1)
9	28	63	196	5	4	1	0	7	(10|1)
9	28	63	197	8	4	3	0	6	(10|1) (16|1)
9	28	63	198	17	3	4	0	4	(10|1)
9	29	64	199	11	5	1	0	6	(17|2)
9	29	64	200	9	5	4	0	5	(10|1) (16|1)
9	29	64	201	4	8	4	0	6	
9	30	65	202	7	5	1	0	5	(16|1)
9	30	65	203	7	8	1	0	0	
10	31	66	204	11	3	5	0	8	(9|1) (15|1) (16|1)
10	31	66	205	6	7	4	0	5	(11|1) (16|1)
10	31	66	206	4	9	1	0	6	(16|1) (17|1)
10	31	66	207	11	6	3	0	4	(17|1)
10	32	67	208	6	4	3	0	7	(9|1)
10	32	67	209	11	8	4	0	2	
10	32	68	210	8	2	2	0	3	(15|1) (17|1)
10	32	68	211	17	5	3	0	0	(10|1)
10	32	68	212	7	4	3	0	5	(10|1)
10	32	68	213	7	3	1	0	6	(9|1)
10	33	69	214	5	5	5	0	5	
10	33	69	215	12	8	5	0	7	
10	33	69	216	5	5	2	0	3	(9|1)

10	33	70	217	4	6	3	0	1	(10|1) (13|1)
10	33	70	218	16	8	1	0	3	(9|1) (17|1) (18|2)
10	33	70	219	5	7	4	0	7	(12|1)
10	33	70	220	6	6	4	0	2	(16|1)
10	33	70	221	6	6	2	0	1	
10	33	71	222	7	4	2	0	3	
10	33	71	223	7	8	3	0	4	
10	33	71	224	8	7	1	0	2	
10	34	72	225	5	4	3	0	5	(10|1)
10	34	72	226	7	7	2	0	8	(9|1)
10	34	73	227	8	8	4	0	7	(10|1)
10	34	74	228	8	3	4	0	8	(9|1)
10	34	74	229	8	0	1	0	3	(9|1)
11	35	75	230	5	3	4	0	4	(9|1) (10|1) (13|1)
11	35	75	231	5	5	3	0	5	(10|2) (15|1)
11	35	75	232	4	5	1	0	8	
11	35	75	233	3	6	4	0	2	(11|1) (16|1)
11	35	76	234	8	5	4	0	4	(10|1) (13|1) (17|1)
11	35	76	235	3	8	3	0	6	(9|1) (10|1) (11|1) (12|2)
11	35	76	236	7	2	3	0	4	(9|1)
11	35	76	237	5	6	4	0	7	(10|1)
11	36	77	238	5	4	1	0	6	(10|1) (13|1) (16|1)
11	36	77	239	2	6	4	0	4	(9|1) (10|1)
11	36	77	240	6	4	4	0	0	
11	36	77	241	2	10	1	0	3	(10|1)
11	36	77	242	3	4	1	0	6	(16|1)
11	37	78	243	3	5	3	0	7	(10|1) (15|1)
11	37	78	244	3	8	4	0	4	(15|1)
11	37	78	245	4	3	4	0	3	(9|1)
11	37	79	246	4	6	2	0	7	(9|1) (10|1) (15|1)
11	37	79	247	2	10	2	0	3	(10|1)

11	37	80	248	5	4	1	0	3	
11	37	80	249	6	4	1	0	7	
11	37	80	250	3	7	1	0	8	
11	37	80	251	3	5	1	0	6	
11	38	81	252	2	6	1	0	7	(15|2)
11	38	81	253	4	5	6	0	5	(17|1)
11	38	81	254	2	7	3	0	8	(10|1)
11	39	82	255	8	6	3	0	5	(15|1)
11	39	82	256	4	6	2	0	4	(9|1)
11	39	82	257	6	6	4	0	8	(9|1) (10|2) (13|1)
11	39	83	258	3	8	2	0	0	(12|1)
11	39	83	259	4	7	3	0	2	(9|1) (10|1) (16|1)
11	39	83	260	9	0	1	0	5	(15|1)
12	40	84	261	5	3	1	0	5	(16|1) (18|1)
12	40	84	262	15	6	2	0	8	(15|1) (16|1)
12	40	84	263	5	1	1	0	1	(9|1) (10|1)
12	41	85	264	11	8	1	0	3	
12	42	86	265	7	3	4	0	7	(9|1) (12|1)
12	42	86	266	5	6	1	0	6	(10|1) (16|1)
12	42	86	267	6	4	3	0	4	
12	42	87	268	6	5	1	0	4	
12	42	87	269	9	4	3	0	1	(9|1) (10|1)
12	42	87	270	5	5	2	0	6	(9|1) (10|1)
12	42	87	271	7	7	3	0	7	(13|1)
12	43	88	272	22	6	1	0	6	
12	43	88	273	16	6	2	0	3	
12	43	88	274	5	1	3	0	8	(10|1)
12	43	89	275	16	6	3	0	3	(10|1) (12|1)
12	43	89	276	10	4	4	0	4	(17|1)
12	43	90	277	7	5	3	0	3	
12	43	90	278	24	3	3	0	6	(9|1) (10|2)
12	43	90	279	11	6	4	0	7	(12|2)

12	44	91	280	5	4	4	0	6	(10|1)
12	44	91	281	5	6	3	0	1	(12|1)
13	45	92	282	5	4	4	0	8	(9|1)
13	45	92	283	6	2	4	0	4	(10|1)
14	46	93	284	5	7	4	0	7	(13|1)
14	46	93	285	3	4	3	0	3	(10|1)
14	46	93	286	4	7	1	0	0	
14	47	94	287	3	4	5	0	4	(10|1)
14	47	94	288	4	6	2	0	8	(10|1)
14	47	94	289	4	5	5	0	7	
14	47	94	290	4	9	1	0	8	(12|2)
14	47	95	291	4	9	2	0	2	(10|1) (14|1)
14	47	95	292	5	5	4	0	7	
14	48	96	293	3	8	4	0	5	
14	48	96	294	10	3	2	0	0	(10|1)
14	48	96	295	3	4	4	0	3	(9|1) (10|1)
14	48	97	296	2	4	1	0	6	(10|1) (12|1) (15|1)
14	48	97	297	5	7	1	0	0	(15|2) (16|1)
14	48	97	298	5	4	5	0	5	
14	48	97	299	5	8	6	0	0	(9|2) (10|1) (16|1)

Table A.23. Routing Structure and Operation Related Data - Problem 6

Operation	Setup times (preceding operation | setup)
2	(6|5) (28|7) (36|5) (41|6) (55|6) (81|8) (106|3) (116|6) (138|5) (147|2) (157|3) (166|6) (190|5) (209|6) (220|6) (224|7) (233|8) (291|5)
6	(2|8) (28|7) (36|9) (41|6) (55|8) (81|6) (89|4) (106|7) (116|1) (138|4) (147|3) (157|8) (190|4) (209|6) (291|6)
28	(2|6) (6|8) (36|6) (41|5) (55|4) (81|6) (89|6) (106|5) (116|4) (138|5) (147|5) (157|5) (166|7) (190|7) (209|5) (220|7) (224|5) (233|8) (259|5)
36	(2|3) (6|6) (28|3) (55|6) (81|4) (106|8) (116|4) (138|6) (147|8) (157|2) (166|9) (190|6) (209|7) (220|4) (224|2) (233|3) (259|6) (291|2)
41	(2|5) (6|7) (28|6) (36|5) (55|5) (81|7) (89|5) (116|5) (138|8) (157|5) (190|7) (209|6) (220|7) (224|5) (233|2) (291|3)
42	(57|10) (64|9) (74|5) (122|9) (137|8) (203|4) (211|6) (258|2) (286|7) (297|6) (299|4)
55	(2|5) (6|5) (28|5) (36|5) (41|8) (81|8) (106|8) (116|0) (157|5) (166|4) (190|5) (209|3) (220|6) (224|2) (233|5) (259|3) (291|6)
57	(42|7) (64|4) (74|5) (122|3) (137|7) (167|5) (211|5) (240|5) (286|5) (297|5)
60	(75|6) (100|3) (105|8) (108|5) (111|3) (117|5) (148|3) (181|6) (189|6) (269|5) (281|4)
64	(42|5) (74|4) (122|3) (137|7) (167|5) (211|6) (240|4) (258|4) (286|4) (297|6) (299|9)
74	(42|4) (57|4) (64|6) (122|3) (137|4) (167|6) (211|3) (240|2) (286|6) (297|4) (299|6)
75	(60|3) (105|2) (108|7) (111|7) (117|5) (148|4) (181|5) (189|4) (269|4) (281|7)
81	(2|10) (6|3) (28|3) (41|6) (55|8) (89|6) (106|5) (116|1) (138|5) (157|6) (166|4) (190|8) (209|6) (220|10) (224|4) (233|8) (291|4)
89	(2|5) (6|2) (28|8) (36|2) (41|8) (55|6) (81|5) (106|7) (116|6) (147|8) (166|7) (209|7) (224|6) (233|5) (259|4) (291|3)
100	(60|10) (75|6) (105|4) (111|6) (117|4) (189|6) (269|4) (281|8)
105	(75|4) (100|4) (108|2) (111|9) (117|3) (148|7) (181|6) (269|3) (281|3)
106	(2|9) (28|4) (36|8) (41|5) (55|3) (89|5) (116|3) (138|2) (147|5) (157|3) (209|8) (220|4) (224|4) (259|5) (291|5)
108	(60|3) (100|3) (105|7) (111|4) (117|4) (181|7) (189|1) (269|0)
111	(60|7) (75|3) (100|6) (105|7) (108|4) (117|6) (148|3) (181|7) (189|3) (269|2)

	(281|6)
116	(2|4) (28|6) (36|2) (41|6) (81|7) (89|7) (106|5) (147|7) (157|7) (166|9) (209|6) (220|4) (224|5) (291|6)
117	(60|4) (75|4) (100|10) (105|5) (108|4) (111|10) (148|2) (181|4) (189|2) (269|4) (281|3)
122	(57|9) (64|6) (137|4) (167|6) (203|8) (240|6) (286|8) (297|4) (299|6)
137	(42|3) (57|7) (122|4) (167|7) (203|4) (258|2) (286|8) (297|9) (299|0)
138	(2|7) (6|9) (28|6) (36|5) (41|7) (55|6) (81|5) (106|0) (116|4) (157|3) (190|6) (209|2) (220|2) (224|7) (233|9) (259|7) (291|3)
147	(2|8) (6|4) (41|4) (55|8) (81|2) (106|7) (116|4) (138|4) (166|5) (190|5) (209|3) (220|5) (224|3) (233|2) (259|5) (291|7)
148	(60|7) (75|8) (100|4) (105|4) (111|7) (117|4) (181|4) (189|6) (269|6) (281|6)
157	(2|5) (6|0) (36|4) (55|2) (81|7) (89|0) (106|4) (116|5) (138|5) (147|3) (166|5) (209|7) (220|5) (233|0) (259|10) (291|5)
166	(6|5) (28|3) (36|6) (41|7) (55|7) (81|1) (89|5) (106|8) (116|9) (138|6) (220|5) (224|9) (233|6) (259|6) (291|5)
167	(42|5) (64|6) (74|0) (122|3) (137|7) (203|3) (211|4) (258|4) (286|4) (299|6)
181	(60|9) (100|5) (105|7) (108|0) (111|4) (117|8) (148|10) (189|5) (281|3)
189	(60|2) (100|4) (105|5) (108|10) (117|3) (148|2) (281|5)
190	(2|6) (6|7) (36|1) (41|7) (55|7) (81|6) (106|5) (116|4) (138|7) (147|3) (157|4) (166|8) (209|5) (224|6) (233|5) (259|4) (291|2)
203	(42|6) (64|4) (74|2) (122|7) (137|5) (167|7) (211|4) (258|7) (286|7) (297|6) (299|7)
209	(2|4) (28|1) (41|4) (55|7) (81|6) (89|6) (106|7) (116|3) (138|7) (147|6) (166|5) (190|5) (224|5) (233|0) (259|3) (291|5)
211	(42|6) (57|7) (64|7) (74|6) (122|2) (167|4) (203|9) (240|7) (258|3) (297|6)
220	(6|8) (28|3) (36|7) (41|2) (81|2) (89|1) (106|3) (116|8) (138|3) (147|3) (166|4) (190|3) (209|9) (224|4) (233|1) (259|5) (291|7)
224	(2|5) (6|3) (28|5) (36|6) (41|5) (81|5) (89|9) (106|6) (116|4) (138|5) (147|4) (157|6) (166|5) (190|6) (209|3) (220|6) (233|8) (259|4) (291|3)
233	(6|4) (28|5) (41|5) (81|5) (106|4) (116|4) (138|7) (147|5) (157|4) (166|5) (190|5) (224|7) (259|3) (291|9)
240	(42|8) (57|5) (64|0) (74|7) (122|2) (137|7) (167|9) (203|6) (211|5) (258|5)

	(286|2) (297|3) (299|3)
258	(42|6) (64|6) (74|6) (137|5) (211|5) (240|5) (299|2)
259	(6|4) (28|3) (55|3) (81|8) (89|5) (116|4) (147|4) (157|4) (166|8) (190|4) (220|3) (224|6) (233|6) (291|6)
269	(60|8) (75|8) (100|5) (105|7) (108|0) (111|5) (117|6) (148|5) (181|9) (281|5)
281	(60|3) (75|4) (100|3) (105|5) (111|5) (117|3) (181|7) (269|5)
286	(42|4) (57|5) (64|4) (74|5) (122|4) (137|5) (167|5) (203|6) (240|5) (258|3) (297|5) (299|3)
291	(2|6) (6|5) (36|0) (55|3) (81|8) (106|3) (116|3) (138|8) (147|2) (157|5) (166|7) (190|4) (209|2) (224|4) (233|2) (259|5)
297	(42|6) (57|6) (64|6) (122|3) (137|6) (167|4) (203|8) (211|6) (240|6) (258|0) (286|4) (299|4)
299	(42|2) (57|4) (64|5) (74|5) (122|6) (137|6) (167|4) (203|6) (211|6) (240|4) (258|7) (286|3) (297|6)

Table A.24. Operations with Sequence Dependent Setup Times - Problem 6

APPENDIX C

COMPUTATIONAL STRUCTURE OF THE HYBRID SCHEDULING SYSTEM

The hybrid scheduling system was implemented in C language. An overview of the computational system structure is shown in the following diagram:

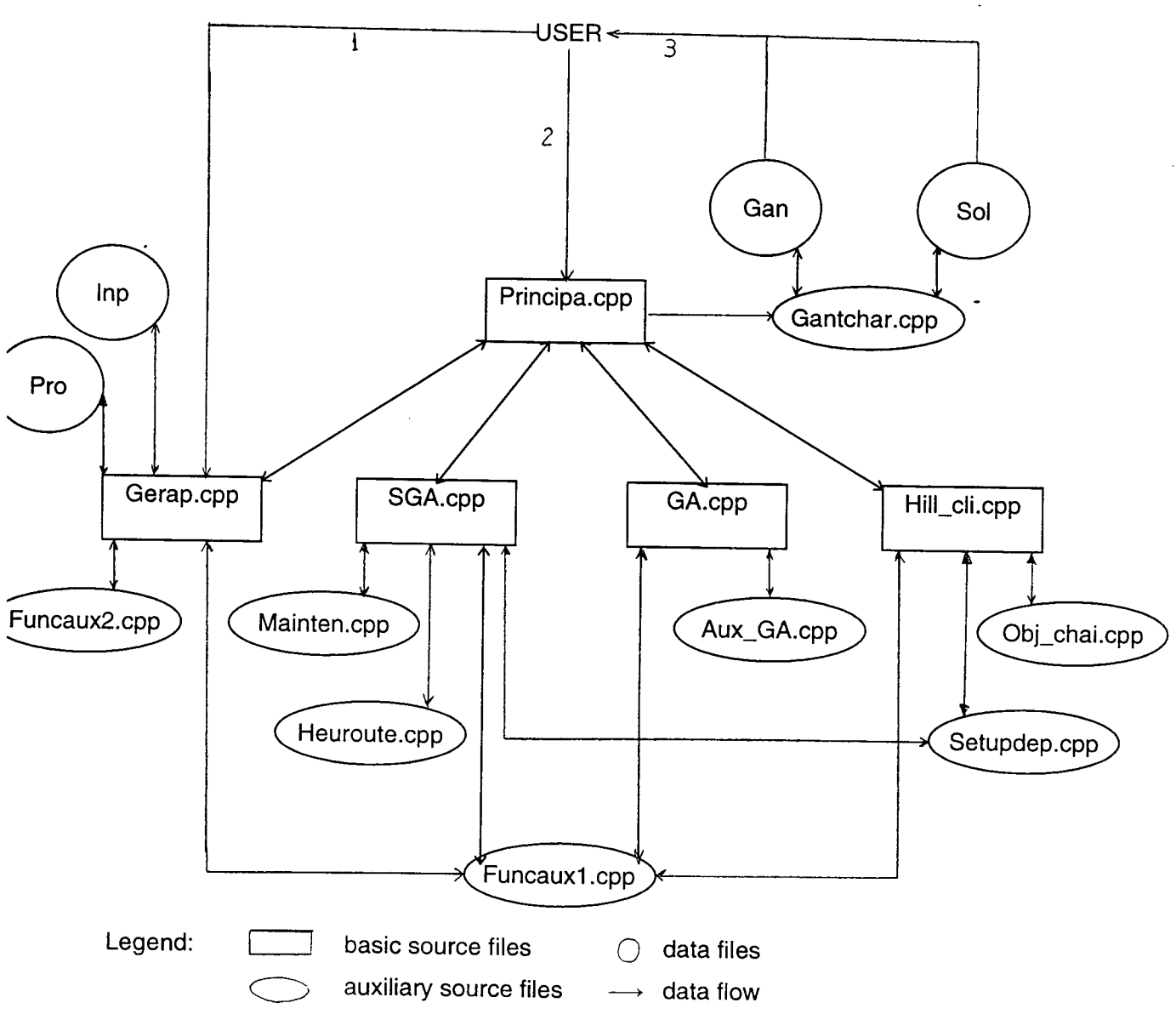


FIGURE C.1. System Structure Diagram

The main function that controls and integrates the entire system is located in file Principa. Functions in Gerap are responsible for randomly reading, generating, or loading a scheduling problem. The modified schedule generation algorithms, the local hill climbing and the genetic algorithm are implemented respectively in SGA, Hill_cli and GA. The other source files contain functions that are used by the basic system modules. The data files (binary and text) store problem and solution data.

The user must select among loading an existing problem from disk, entering a new problem through the keyboard, or using the random problem generator described in appendix B (interface 1 in the diagram). These functions are implemented in Gerap. After the selection the user must provide the problem data or the inputs requested by the problem generator. Inp and Prob are a set of binary and text files that store problem instances like the ones described in appendix A. Funcaux2 contains some functions called by the problem generator (e.g., functions that randomly generate values from a probability distribution).

Once the problem is defined (and loaded), the user chooses the system configuration (interface 2 in the diagram). Precisely the user determines the following parameters:

1. Basic system configuration:
   - only the schedule generation algorithm
   - schedule generation algorithm followed by a local hill climbing
   - schedule generation algorithm followed by a genetic algorithm hybridized with a local hill climbing
   - schedule generation algorithm followed by a pure genetic algorithm

2. Schedule generation algorithm:
   - active or non-delay
   - route selection method: random selection or route selection heuristic
   - parameter 'a' in the route selection heuristic (if it is selected)
   - dispatching procedure: randomly or SPT rule
   - resource selection method: randomly or minimum gap heuristic
   - generation mode: complete or simplified

3. Local hill climbing:
    - method for selecting the arc to be reversed: random choice or bottleneck heuristic
    - resources are allowed or not allowed to change when a move is performed

4. GA parameters:
    - population size
    - number of generations
    - basic crossover rate
    - basic mutation rate
    - route crossover rate
    - route mutation rate
    - decoding strategy: active or non-delay

5. Coefficients a_i and B of the multiobjective evaluation function F = a_1.mean completion time + a_2.mean tardiness + a_3.makespan + a_4.maximum tardiness + a_5.maximum lateness + a_6.maximum flow time + B

Note that any other multiobjective function can be easily implemented. But as described in chapter 4, most regular measures are equivalent to one of the single criteria included in function F.

According to the basic system configuration chosen by the user the main function calls the proper sequence of functions that run the schedule generation algorithm (in file SGA), the local hill climbing (in Hill_cli) and the genetic algorithm (in GA).

The decoding procedure used by the GA is the modified schedule generation algorithm. The GA can also be hybridized with the local hill climbing procedure. Therefore there exist data flows between GA and SGA, and between GA and Hill_cli (see diagram C.1).

The final solution is stored in the output files Gan (Gantt chart) and Sol (numerical data) with the help of functions located in file Gantchar. Gantt charts were used to aid the

verification of the proposed models, i.e., the consistency of the solutions provided by the hybrid scheduling system were verified using Gantt charts.

The auxiliary file Funcaux1 carries some functions called by almost all the basic files, like functions that return the successor or predecessor of an operation, subprocess or part, functions that calculate overlap times and earliest start times, etc. Functions that manipulate setup information (e.g., update setup times after a neighborhood move) are implemented in file Setupdep. Functions concerning to scheduled breaks of resources are implemented in Mainten. The route selection heuristic is stored in file Heuroute. Some GA related function and some Hill climbing related functions are implemented in files Aux_GA (e.g., selection procedure) and Obj_chai (e.g., determination of the active chains). The total number of code lines of the system reaches 9189.

APPENDIX D

PERFORMANCE OF THE HYBRID SCHEDULING SYSTEM ON CLASSICAL JOB SHOP SCHEDULING PROBLEMS

The performance of the proposed model on for Taillard's benchmarks is described here. All the three modules of the hybrid system (the modified schedule generation algorithm, the local hill climbing and the genetic algorithm) were set to their basic configurations.

For each problem thirty runs were carried out. Table D.1 compares the performance of the proposed system with the performance of the hybrid genetic algorithm GA3 proposed by Mattfeld. Notice that GA3 has yielded the best solutions for Taillard's benchmarks among all the GA reported in the literature. The number of runs performed by Mattfeld's GA3 for each problem was also thirty. The population size and number of generation were fixed at 100 (the same used in our system).

Problem (nº jobs x nº machines)	Best solution Mattfeld's GA3	Best Solution Proposed Model	Difference (%)
ta01 (15x15)	1247	1299	4.0
ta11 (20x15)	1411	1458	3.2
ta12 (20x20)	1723	1736	0.7
ta13 (30x15)	1813	1902	4.7

Table D.1. System Performance on Taillard's Job Shop Problems.

Figure D.1 shows the behavior of the hybrid system over generations for each of these classical job shop scheduling problems. Only the best among the thirty runs is plotted.

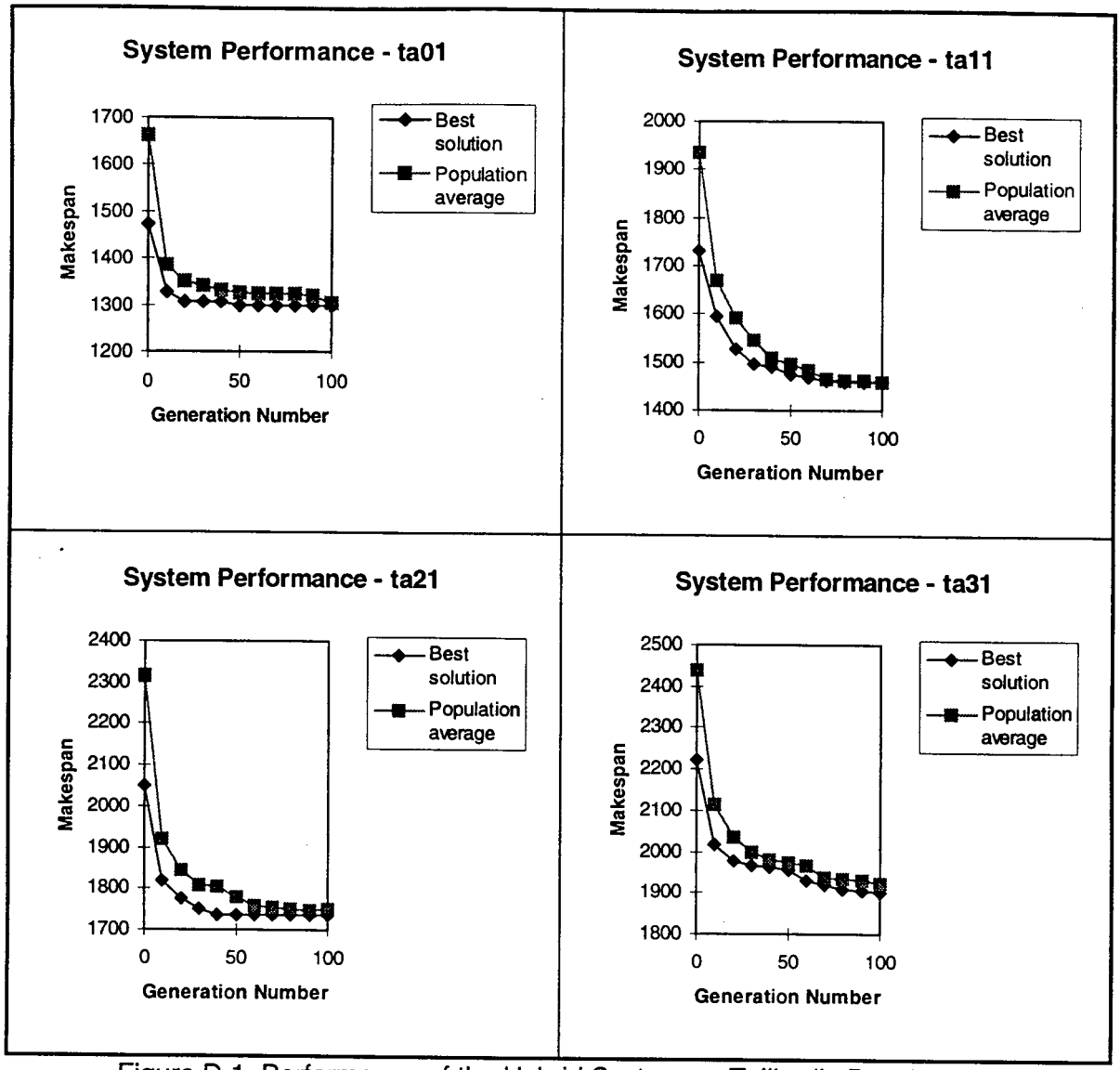


Figure D.1. Performance of the Hybrid System on Taillard's Benchmarks