Reinforcement Planning for Resource Allocation and Constraint Satisfaction

Bing Liu

PhD

University of Edinburgh

1988

Abstract

Planning or scheduling involving resources and constraints is a difficult task. Past research has only had limited success. In this research, job shop scheduling is used to further investigate the problems.

Essentially, job shop scheduling is the allocation of different resources to satisfy various shop constraints (or meet the shop needs). In order to understand its underlying problems, a detailed analysis of the scheduling process, within an AI framework, is proposed. This highlights the deficiencies of the current techniques and suggests that it is the limited resources which restrict the satisfaction of constraints, as there are typically conflict situations between constraint satisfaction and resource availability. The central problem in dealing with resource allocation is how to predict what effects the decisions currently being made will have on the following scheduling or how to have a global perspective of the scheduling process. For this purpose, "reinforcement scheduling" is introduced. This method builds a utilization plan (reinforcement plan) as a basis for generating decision making guidelines for constructing the detailed schedule.

However, in any situation, there is normally more than one constraint that needs to be considered. If the resource is left as it is (since the resource capacity cannot be changed), there will typically be conflict situations among the constraints themselves. These conflicts also hamper the satisfaction of constraints. To deal with them, a global perspective is still the key, and the reinforcement approach helps, but it is not sufficient. With many constraints to be satisfied in scheduling, we argue it is very ineffective to try to consider all of them in a single process. Therefore, a constraint decomposition approach is suggested. Instead of trying to satisfy all the constraints in a single scheduling process, this approach will consider different constraints in different processes. In each process, the reinforcement technique will play an important role.

Within these two approaches, scheduling becomes a multi-stage process where each stage can be seen as a complete scheduling.

After a schedule has been constructed, it will be used to control the production process. In real situations many unexpected problems may occur, which may force changes to be made to the existing schedule. Discussion of the problems shows that what is needed here is an analysis of the current situation and overall situation, and a consultation of the existing schedule to combine them to make decisions. This is what the reinforcement approach provides. So, reinforcement scheduling also forms the central technique in rescheduling.

Based on both of these approaches, a scheduling system called RESS-II (REinforcement Scheduling System) has been implemented. It includes both the construction of a schedule and the maintenance of it.

Acknowledgments

I wish to express my deep thanks to my supervisors, Austin Tate and Ken Currie, for their guidance and helpful comments throughout my research. They have not only given me invaluable technical supervision, but also have been so patient in reading my thesis draft and other papers several times to correct the mistakes, and this has helped me a great deal in improving my English. Many of the mistakes (both technical and language) and vagueness in the earlier drafts of this thesis have been eliminated due to their meticulous and critical reading. I am very grateful to them for all their help. I am also indebted to Jim Howe who is also one of my supervisors for his support and encouragement to this research.

James Kwa, Mark Drummond (AIAI) and Chee-Kit Looi have also kindly consented to proofread the chapters. I wish to thank them for their suggestions and comments which have helped to clarify various parts. Many other people in the department have also helped in many ways, for which I am indeed grateful.

There are also many friends outside the department who have helped my study and made my stay in Scotland such a memorable time. My gratitude goes particularly to Isabelle Chaignepain and Ken Gourlay (Stevenson College).

This work was supported by the Chinese Government, the British Council and an ORS award. I am grateful to the Education Section of the Chinese Embassy in London. They have been so understanding and have given me all the help I needed.

Declaration

I declare that this thesis has been composed by myself and that the work it reports is my own.

Contents

Chapter 2

Chapter 3

List of Figures

Chapter 1

Introduction

This research is directed towards dealing with the problem of resource allocation and constraint satisfaction in AI planning or scheduling, i.e. how to allocate resources and how to satisfy constraints. However, the aim is not to attempt to include all kinds of resources and constraints in the planning or scheduling process. Instead, it is to study how available resources can be effectively allocated to actions in order to best satisfy the constraints, i.e. how to construct efficient plans or schedules in the situations where resources and constraints are involved.

Past research in AI planning (or scheduling) and problem solving is mostly confined to the causal action ordering aspect of the problem, i.e. detecting and correcting interactions between actions. Most planners do not handle resources and constraints. The ones that do have had only limited success. The resource handling techniques developed to date can only check the resource availability against partial solutions rather than allocate them to the actions in an efficient way. However, in most of the realistic problem solving domains, resources and constraints are involved and in such situations efficient plans or schedules are often required. If planning is to be more realistic, the demand for better resource allocation and constraint satisfaction becomes acute.

In this thesis, we will further analyze these issues based on the domain of job shop scheduling, which is an outstanding problem of resource allocation and constraint satisfaction. To solve the underlying problems two new approaches are proposed, a reinforcement scheduling approach and a constraint decomposition approach.

In order to fully appreciate how this research furthers the cause of resource allocation and constraint satisfaction in planning and scheduling, it is first necessary to know what has been done in the past. The following section is a brief review of the major planning work (it is limited to AI planning research only, other types of knowledge-based systems will be reviewed in the next chapter) that has preceded this research.

1.1 A Historical Perspective of Planning

Planning can be defined as the predetermination of a course of actions aimed at achieving specified goals. Its problems can range from trivial sequential plans to highly parallel plans. Generally, there are two stages in planning: plan construction, and plan execution monitoring and replanning. To date many planners have been built and many planning techniques have been developed, but basically, they fall into two planning paradigms, namely, the state based approach and the least commitment approach.

1.1.1 State-Based Approach

Perhaps, the earliest systems in planning and problem solving are GPS [1] and STRIPS [2]. They are state-based planners. GPS (General Problem Solver), due to Newell, Shaw, and Simon, introduced the means-end approach to problem solving. In this approach, the operators which will best reduce the difference between the current state of the world and the desired goal state are applied. In other words, only those operators that can satisfy some outstanding goals may be considered.

A problem in GPS was presented as:

- A description of initial world state
- A description of goal state
- A function for measuring the difference between states
- A set of operators or actions
- An operator-difference table

Planning or problem solving in GPS is about finding a sequence of operators which will transform the initial world state to the goal state. This process starts by finding the difference between the goal state and the initial state. If no difference is found, then the problem is solved. Otherwise, using this difference, GPS looks in the operator-difference table for operators which are indexed under a difference that has been formed. It then applies the operators to the initial state, to produce a new state. Sometimes, the operators have preconditions that have to be met. Then, the system will use these preconditions, or a state of the world that has these preconditions satisfied, as a goal state. Such recursion happens as necessary. The resulting new state is measured against the goal state. New operators are applied until no difference exists.

There are some serious problems with GPS. One is the difficulty in translating a problem into the search formalism. Another is that the differences between states are sometime difficult to find. When a problem grows large, it will also require a huge difference table. As this is prepared by the user, a great burden will be put on such a user.

STRIPS [2] is a improvement to the GPS system. It essentially uses the same search approach as GPS, but has a standard way of describing the state of the world. In STRIPS, the world is represented as a conjunction of facts in first order predicate calculus. This makes finding the difference much simpler as the facts in the goal state, but not in the current state, are the differences.

Operators were also standardised to perform only two functions; adding facts to the data base and deleting facts from the data base. An add-list and a delete-list are associated

with each operator as the effects on the state of the world caused by applying the operator. So, when an operator is applied, the facts in its delete-list are deleted from the data base and the facts in its add-list are added to the data base. The resulting data base becomes a new state of the world.

As the effects on a world state are represented in each operator, there is no need to have an operator-difference table. When a difference is found an operator which has the difference as one of its add-list effects is selected to be applied.

Backtracking occurs when a wrong operator is applied. The reason for this is that there may be more than one operator that can be applied to a state and there is no prior basis for selecting one. To backtrack, the system finds a point in its planning process where it had a choice of more than one operator to apply, and this time applies a different operator. Such backtracking arises from premature commitment to a problem-solving path and can be combinatorially explosive. Later planners have been designed to avoid such pre-commitment as much as possible.

1.1.2 Least Commitment Approach

Before researchers discovered the least commitment approach to problem solving, there were some other approaches that appeared. Sussman's HACKER [3] system solved blocks world problems with a conjunction of goals by first making some assumptions to form a plan, which was then debugged. To form a plan in HACKER is actually to search for a plan procedure known to be appropriate for the problem in its solution library. If one is found but it does not achieve the goal required, it debugs this plan using programs called "critics", which can recognize some interferences. The bugs found will be patched to make the plan work. To patch the bugs, HACKER reorders the goals. But a big problem with HACKER is that it cannot reorder the goals at different levels of a plan, which may result in many redundant steps in achieving a goal. Furthermore, debugging in HACKER can only handle very minor problems.

Tate in his INTERPLAN system [4] introduced another technique called "Goal Structure". By using such a table describing the "holding period" for goals and subgoals over which they can be assumed to be true, INTERPLAN can detect interactions and reorder the goals or subgoals to ease conflict situations. At the first instance after an interaction is detected, like HACKER INTERPLAN reorders the subgoals at a single level. If it fails, INTERPLAN can use the goal state to propose minimal re-ordering of the subgoals at different levels of the plan to avoid interactions.

HACKER backtracks after it finds the interactions or protection violations. Such backtracking will reorder the goals and subgoals to avoid the interactions. But when the problem grows large this method is very inefficient. As this technique is very destructive, the same goal can be achieved over and over again. An alternative approach is proposed by Waldinger [5], called "goal regression". The main idea is that if there are several goals to be achieved, a planner will first achieve one goal and then in solving the others, the planner regresses the latter goals to earlier parts of the plan by adding more instructions to achieve these goals while also maintaining the original one. The advantage is that this approach is constructive. The debugging of the "approach" to solving a problem in INTERPLAN is also constructive.

The first least commitment planner was built by Earl Sacerdoti in 1975, called NOAH [6]. It was later improved by Tate in his NONLIN system [7]. NONLIN had a backtracking top-level control structure added to NOAH's set of plan modification operations, so that it could find plans after NOAH would get stuck.

In a least commitment approach, a plan is represented as a partially ordered action network. This network is a graph where nodes are tasks, subtasks, and primitive actions, and links are ordering relations. The links are classified as plan links, subtask links and ordering links. Such classification expresses the possible hierarchical structure of the planning process.

Problem solving in least-commitment planners seeks to develop such an action network following a hierarchical expansion (although the state-based planners can also plan in a

hierachical fashion). From a single action to be performed, a hierarchy of nodes are developed to more and more concrete descriptions. After the expansion of the original action, there will be two levels of representation. The first is the original action and the second is the subgoals, a more detailed desccription of how it can be performed. These lower level actions can be further expanded to more detail, and such expansion continues until finally a level of subgoals can be attained by primitive actions that can be executed directly.

The fundamental idea behind node expansion in least commitment planners is the following. To achieve a conjunction of goals **G1, G2, G3, ... Gn**, the planner finds a set of subgoals **S1 = a1, a2, a3, ... an** that can lead to the application of an operator which will achieve **G1, S2 = b1, b2, b3, ... bn** that achieves **G2** and **Sn = n1, n2, n3, ... nn** that achieves **Gn**. These subgoals can be further achieved if necessary in a hierarchical fashion. When a goal node is expanded to subgoals in order to achieve that goal, the planner runs the risk of creating interacting problems. To solve these problems, the planner works in two ways; first, by not ordering the subgoals until there is some reason for it and, second, by continually detecting interactions and correcting them at each level of expansion. This approach is termed a least commitment method, which means that ordering is only introduced when it is necessary or when an interaction is detected and such ordering can resolve the interaction. It contrasts with the earlier planners such as STRIPS, which order the operators (or actions) arbitrarily even if there is no ordering relationship between them. If a problem is found they will have to backtrack and replan to avoid the interaction. A great deal of backtracking can be avoided in least-commitment planners.

Like STRIPS, NOAH keeps track of the facts that are added and deleted by its action. These facts and effects are stored in a 'Table Of Multiple Effects', TOME. The TOME can be manipulated by a set of "critics" to detect interactions and correct interactions. NONLIN also introduced a Goal Structure table (GOST) to register the scope of effects as used by conditions in the plan.

1.2 Resource Allocation and Constraint Satisfaction

1.2.1 Resource Allocation

Apart from the ordering relationships among actions discussed above, in most real world planning and scheduling problems, various kinds of resources also exist. They restrict the planning process in that the actions must have their required resources available when they are planned or scheduled. Otherwise, execution will not be possible. That is to say a planner not only has to order the actions so that they are free of interactions but also to consider resource availability at the same time before planning and scheduling can be performed. They create one more problem dimension for planning (in addition to the action ordering).

Normally, in resource-restricted domains, the total amount of resources available and/or the total capacities resources have over a time period are limited. Execution of one action will typically have some influence on the execution of the others. In other words, these actions compete with each other for the same resources. For instance, consider the case where there are two actions that can be executed in parallel legally (which means there is no interaction between them) and both require the same resource. Assume there is only one unit of this resource available and it can only be used by one action a time. So, these two actions have to be performed successively (it does not matter which action is executed first if neither of the sequences affects legitimacy of the plan and if there is nothing to constrain the result), rather than in parallel. This sequence of execution creates an extra link between these two action, and this link has nothing to do with their causally required ordering sequence. We call such a link sequencing link. So, in resource limited domains, besides ordering links between actions, there are also sequencing links in a plan.

Although both of these kinds of links express a precedence that one action is executed before and/or after and/or at the same time with other actions, their underlying meanings are quite different. The sequencing links are not so rigid as the ordering links,

which means that the sequence of actions that are caused by limited resources can be changed. In other words, rearrangment of the sequencing links does not affect a plan's causal legitimacy if the ordering links are still there. The combination of these two makes planning more complicated. Let us take another example. Suppose a plan is required for the blocks world problem shown in Figure 1.1 (A). The robot which is going to execute the plan has only a limited number of arms. (B) shows a plan for the robot that has only one arm and (C) shows a plan for robot that has two arms (T represents Table). In (B), the link between action [put A on T] and action [put D on T] is a sequencing link. If action [put D on T] is put before action [put A on T], the plan is also workable. Note however that in the latter case, action [put A on T] can be put anywhere before action [put B on C], while in the former as shown in (B), after action [put A on T] is executed action [put D on T] has only one position but nowhere else.

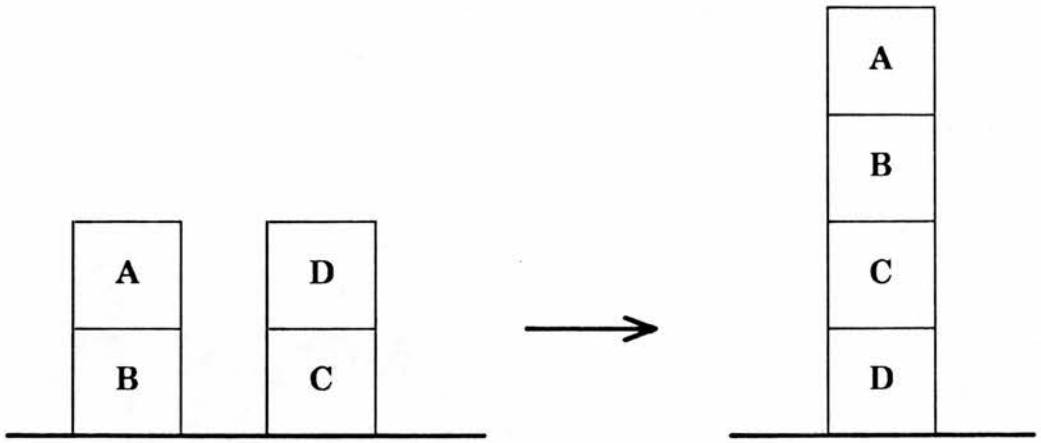
There could be different kinds of resources as identified in [8], and they typically create different problems.

- Consumable resources

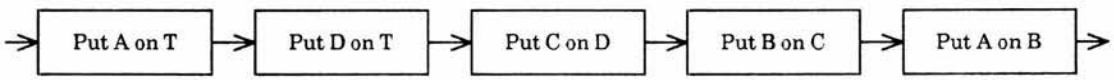
  These resources are the ones for which an initial stockpile is available, which can only be used up by actions. In managing such resources, the time when they are used is not important. Only the total usage is relevant and one must ensure that this does not exceed the initial availability.

- Shared resources

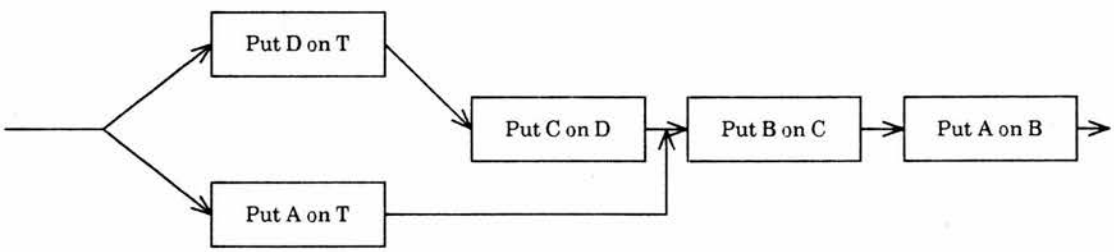
  These resources are shared by various actions, but are not consumable. Normally, they have fixed capacities over the time period concerned. Allocation of these resources actually specifies when each of them will be used by which actions. Typically, if a resource is occupied by an action during a time period, it cannot be allocated to another action. If the capacity of a resource is not used during a period of time, then it is lost and cannot be got back as time progresses. This is very different from the consumable resource. Machines in a job shop are a good example of a shared resource.



(A) Initial and Goal State



(B) A plan for robot with only one arm



(C) A plan for robot with two arms

Figure 1.1 Resource in planning

- Renewable resources

  A renewable resource is some resource that is required in some amount by some actions and is produced in various amounts by other actions or by external agents. It can be consumable as well as shared. Money is an example of a renewable resource: An action's task will use up the supply of the resource (eg. spending some money), while other actions increase the supply (eg. by earning money).

If a domain has resources, but they are sufficient so as not to constrain a plan, the planning and scheduling process will almost be the same as for normal planning. Only one step needs to be added, that is to check resource availability. If a resource is consumable and it is used up by a partial plan, then the goal may not be reached, although there may be a plan that exists and can lead to the goal if some extra resource is provided. This does not mean that there is no plan within the resource available either as this partial plan may not be efficient (finding a plan with the resource available is also regarded as a restriction). If the resource is a shared resource, delaying the execution of actions may ensure a complete plan.

However, in normal situations, an efficient plan is often required, and the amount or the capacities of resources available are limited. So, resources will typically restrict a planning process. As an efficient plan is often expressed as constraints to be satisfied, in the next section, we will introduce the constraint satisfaction problem.

1.2.2 Constraint Satisfaction in Planning

A constraint is a requirement that the planning process needs to meet. It not only represents a measurement for the final planning result, but also imposes further restriction on the planning process. Such restriction is, however, different from that imposed by limited resources. A resource restricts a plan through its finite capacity and total amount available (i.e. without the necessary resources, an action cannot be executed), while a constraint restricts a plan by putting a demand on the plan and the

planning process has to constantly consult this demand in order to reach a final plan that will meet such demand. That is to say the planning process is to satisfy the constraints, or constraint satisfaction is the objective of the planning. In a general sense, a resource is for the planning process to allocate, while a constraint is for the planning process to satisfy.

A constraint also differs from an ordering relationship between actions. An ordering expresses a causal (or logical) relationship between actions, and if it is not met or an interaction is not corrected the plan produced will not be legal, while a constraint only expresses a desired (or preferred) plan result, and it does not affect the legitimacy of a plan. For instance, in Figure 1.2, a robot needs to visit three places. Since it has to get the payroll from the bank to deliver to the factory, the robot has to go to the bank to get the payroll first before it can deliver it to the factory. This is an ordering (causal) relationship between two actions, 'go to bank' and 'go to factory'. If a traveling time limit (130min) is imposed on the robot within which it should visit all these places, then this time limit is a constraint. In some case, a constraint can be relaxed, which means that a plan that has not satisfied the constraint may also be acceptable, such as in scheduling where the due date may not be satisfied for every order. But an ordering can never be relaxed.

In any domain, there could be many legal plans (a set of plans), but the ones that satisfy the constraints are just a subset of those, or possibly a null set. Traditional planning only looks for one plan, that is any legal plan, while planning involving constraints looks for a plan in the subset. A plan may be perfectly legal but it may take so long to execute that the time limit for the execution is exceeded. This deadline turns a problem that formerly could be solved by any solution into a problem that requires an efficient solution. In most real situations, efficient solutions are often required.

In many cases, the sequence in which the actions of a plan are executed can make all the difference between an efficient plan and one that cannot satisfy the constraints. Imagine the situation shown in Figure 1.2. The traveling time limit for the robot that must visit three places is 145 min. The simplest path is to go from home first to the laundry, then to

the bank, then to the factory and then come back home. The total time required is 140 min. So, the constraint is satisfied and the plan is perfectly legal. However, if the robot goes to the bank first, everything will be changed and there will be no way to satisfy the time limit constraint.

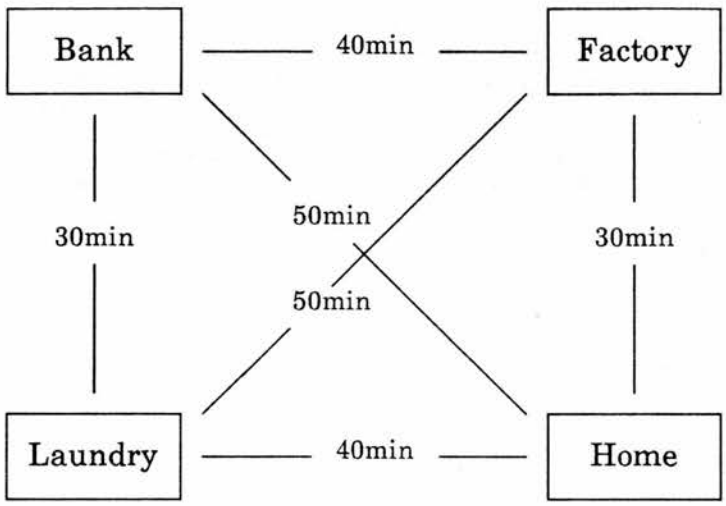


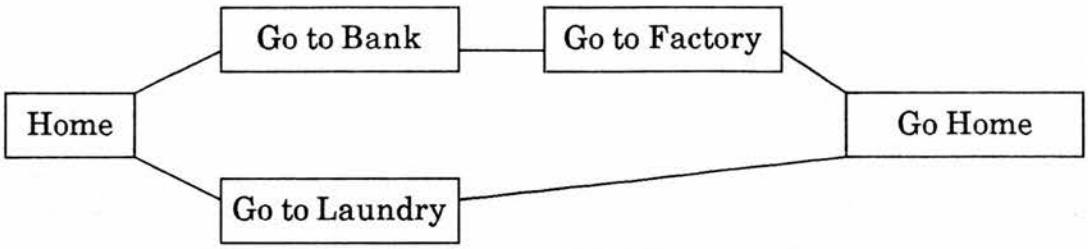
Figure 1.2 Places to be visited in a limited time

Constraints can also be classified. Basically, there are two kinds: global constraints and local constraints. Global constraints, such as a deadline, express the desired final result. They do not care what has happened in the course of the actions. Only the final result counts. For example in scheduling, it does not matter how each operation in an order was scheduled during the schedule construction process. The finish time of its last operation tells whether the due date constraint of the order has been satisfied or not.

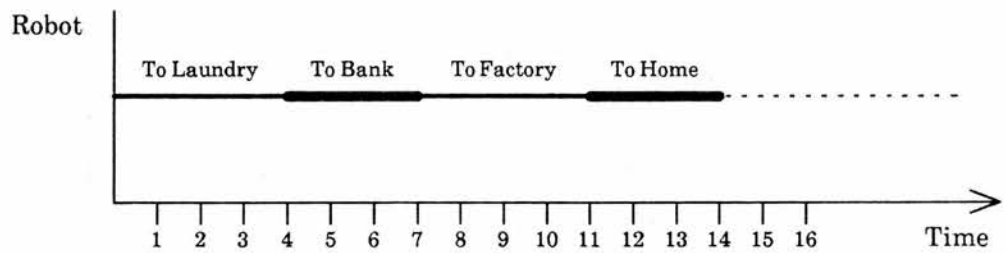
Local constraints express the preferred choice among alternatives at a local choice point. Their satisfaction will normally improve a plan or schedule in one aspect or another.

As mentioned at the end of the last sub-section, resource and constraint problems are normally associated with each other. In such cases, the difficulty becomes even greater, as the problem is unified, i.e. the allocation of resources is for the satisfaction of

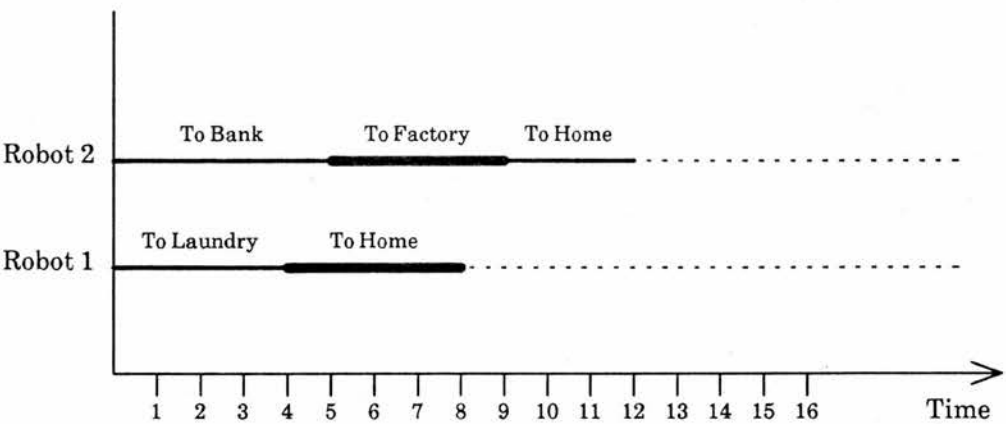
constraints. The above robot errand running gives a good example. The robot is actually the resource.



(A) The plan without resource



(B) A plan with only one robot and with time limit



(C) A plan with two robots and time limit

Figure 1.3 Planning with resource and constraint

If the robot resource is not considered, a plan could be built like the one shown in Figure 1.3 (A), in which action [Go to Bank] and action [Go to factory] are in parallel with action [Go to Laundry]. But let us assume there is a robot to actually perform the task. There will be many possible paths to visit those three places. If there is no time limit involved, the robot could go in any sequence as long as at the end the task is finished. But with a time limit constraint, there is only one path that could lead to a satisfactory result. This has been mentioned above and is shown in Figure 1.3 (B). If the number of robots is two instead of one, the number of possible paths will be greater. Figure 1.3 (C) shows one of the possibilities.

In the next section we will use a realistic domain, job shop scheduling, to further demonstrate the complexity of the problem and how it may be dealt with.

1.3 Job Shop Scheduling as a Problem of Resource Allocation and Constraint Satisfaction

Generally, a scheduling environment can be described as: suppose that it is necessary to execute **q** tasks, **T1, T2, T3,** ... **Tq**, and each task **Ti** consists of a number of subtasks, **t1, t2,** ... **tn**, where each subtask **ti** has a stipulated starting time **si** and finish time **fi** and where each subtask requires a machine **Mi** and a number of tools **Ri**, **Rj** ... **Rm**; to perform these subtasks we have **k** resources; each resource can only perform one subtask at a time, but can execute any number of subtasks in succession. In the job shop scheduling domain, these tasks are the orders (i.e., orders for products; not ordering relations); the subtasks are the operations in each order and resources are machines.

As job shop scheduling is an industrial management function, inevitably, its objective is to maximize profit or reduce cost. This is the constraint. But in practice, it is very difficult to assess the cost of each individual activity within an organization. This objective is broken into many small objectives (or constraints), such as meeting due date, reducing work-in-process time, maximizing resource usage, etc. The constraints

involved in scheduling may include (note, this is a shortlist; a full discussion will be given in the next chapter):

(1) Objective constraints:
Due date
Work-in-process
Production level
(2) Preference constraints:
Set-up constraint
Machine preference

The resources involved in job shop scheduling are:

(1) Machines
(2) Tools.

So, job shop scheduling can be formalised as the allocation of different shop resources in order to satisfy various constraints. The allocation of the resources will ultimately decide the satisfaction of the constraints.

There are two outstanding problems in scheduling that need to be addressed.

(1) Computationally NP-hard

Most of the resource and constraint problems are NP-hard. Imagine the sequencing of just 10 orders with five operations each, and each operation associates with a single machine (no alternatives). Assuming there is no time gaps in the schedule to be generated. There will be $(10!)^5$ or about 10^{32} possible schedules. Situations within a real factory are much more complex. The number of orders, operations and resources are substantially greater. So, the optimal result is often impossible for a computer to reach. In practice, a feasible and approximate schedule is often sought.

(2) Global perspective is essential

Since the permutation approach is not possible, decision making has to be introduced whenever a choice needs to be made. And, in order to have a good feasible schedule, good decisions are essential. As the planning/scheduling of the earlier actions/operations typically influence the planning/scheduling of the later actions, good decisions need effective prediction of the impact of the current decision on following planning. Otherwise, follow on planning may not be possible and the final result could be disastrous.

In the scheduling task, although resource allocation and constraint satisfaction are critical, the ordering relationship between operations is not very complicated. There is only a precedence restriction which states that one operation is to be performed before or after other operations. So, there will be no problem of detecting interactions between operations. (For simplicity, later we will also call this a precedence constraint.)

Once a schedule has been constructed, it will be used on the shop floor to control the production process. According to this schedule, shop activities will be planned and the whole production process will be carried out. However, in real world situations, things may not proceed precisely as planned or scheduled. Many unexpected problems may occur during actual manufacturing, machines may break down, operations may be started and finished later or earlier than scheduled. These problems may invalidate and force changes to be made to the existing schedule. To keep up with the progress of the production process, monitoring and rescheduling is also very important. It further complicates the real world scheduling tasks.

1.4 Reinforcement Approach To Scheduling

In this research reinforcement scheduling and the constraint decomposition approach have been proposed to deal with the resource allocation and constraint satisfaction problem in scheduling. Based on these approaches, a scheduling system, called RESS

(REinforcement Scheduling System), has also been implemented. To date, two versions have been built, but this thesis concentrates on the second version, RESS-II.

1.4.1 Reinforcement Scheduling

Through analyzing the scheduling process within an AI framework, two levels of reasoning activities can be identified, the choice generation level and the choice making level. Choice generation is the logical aspect of finding waiting operations, and this is not very difficult in the scheduling domain. But choice making is actually a resource and constraint restricted decision making, which is the key to scheduling.

Obviously, the criterion for making such a decision will be the satisfaction of constraints, which is also the objective of the scheduling process. But, many of the constraints, such as due date and work-in-process, express preferred final results and many of the problems only manifest after some or all the scheduling has been done. It is almost impossible to access their satisfaction situations at a local state. In problems like scheduling, it is the resources that restrict the satisfaction of constraints. There are typically conflict situations which exist between constraint satisfaction and resource availability. In order to achieve good constraint satisfaction these conflict situations should be avoided or their effects should be maximally reduced. So, at this decision making level, the system must try to choose an operation to allocate the resources in such a way that it will help to lead the scheduling towards a conflict free situation. Then, first of all the system needs to effectively predict when and where these possible conflict situations may occur. In other words, it needs to predict what impact the decision currently being made will have on the following decisions. Hence, besides considering the present situation, this decision making level will also have to take into account the overall situation. But the problem is how to get an overall perspective at a local decision making point.

To deal with the underlying problem, "reinforcement planning" is proposed. The essence of this approach is that the system chooses a critical resource and builds a rough

utilization plan (reinforcement plan) as a basis for generating decision-making guidelines for constructing detailed schedules. There are two scheduling processes (stages) that exist in such an approach; a reinforcement scheduling process and a detailed scheduling process.

The detailed scheduling takes into account the less important resources. Instead of modifying the reinforcement schedule to produce a detailed schedule, the system schedules again. In the process it will try to reduce the effects of the global resource problems by taking extra care, i.e. taking earlier actions or trying to use alternatives, while also dealing with the local problems encountered. It is a process of repeatedly analyzing the current situation by searching through the partial detailed schedule and consulting the reinforcement schedule to make decisions. The search through the partial detailed schedule gives a current situation report on various resources. The search in the reinforcement schedule finds out if any possible problems exist both now and in the future and sees whether they can be solved by proper allocation of resources to the operations. Associated with these searches is an evaluation mechanism to make the final decisions. After a particular decision is made and resources have been allocated to the selected operation, the system updates the reinforcement schedule to make it also reflect the current detailed scheduling situation and more reliable for future analysis.

1.4.2 Constraint Decomposition Approach

Along with the conflict situations between constraint satisfaction and resource availability, there are also conflict situations among constraints themselves. They also hamper constraint satisfaction. Like the conflict situations between constraint satisfaction and resource availability, it is also very difficult to predict and deal with these problems at a local state as most of them only appear when some or all the scheduling has been done. A global perspective is still the main requirement. The reinforcement approach will help, but it is not sufficient. The detailed analysis of constraint classification and scheduling situations suggests that as there are many constraints that need to be satisfied in scheduling, to consider all of them in a single

process is very ineffective. Hence, the constraint decomposition approach is suggested. This method tries to consider different constraints in different scheduling processes. With the reinforcement scheduling at hand, the later processes will be able to predict the future problems at a local state as well as maintain the previous constraint satisfaction results. These two approaches do not conflict with each other; instead constraint decomposition supports the reinforcement approach and the reinforcement approach makes constraint decomposition possible.

1.4.3 Reinforcement Scheduling Vs Complete Search

As scheduling is a NP-hard problem, a complete search for a problem of reasonable size is practically impossible. So, reinforcement scheduling does not do a complete search, and consequently it does not guarantee the best result.

However, neither does it simply cut down the search space so that it can be searched within a feasible time. Instead, it advocates the idea of predicting the impact of local scheduling decisions on an overall solution. Its underlying belief is that not all the resources will influence a schedule very much all the time and there are typically some resources that cause problems during certain periods of time or during the whole time period. If these problems can be predicted and earlier action can be taken to prevent them from happening or reduce their effects, the scheduling result produced will certainly be good. There is no evaluation criteria to judge what schedule is acceptable and what is not acceptable in such a process because such criteria are very difficult to establish. This is because there is no way of finding the best result to decide what can be done. Each scheduling result depends on its individual situation and it cannot be assessed by a common criteria. As a result, most of the constraints can be relaxed during the scheduling process, i.e. they are not necessarily satisfied.

1.4.4 Resource Centered Schedule Representation

The schedule representation in RESS-II is quite different from the traditional AI plan representation. In traditional AI non-linear planners, a plan is represented as a partially ordered action network. The purpose of planning is to build such a complete network, free of interactions. According to this scheme, plan representation is based on actions and their orderings.

However, the schedule representation in RESS-II is a resource schedule, which means that scheduling activities are based on the availability of resources. In performing a scheduling task, the first thing to do is to find a machine which is ready to perform an operation, then select an operation to be scheduled from the ones which are queuing at the machine.

1.4.5 About the RESS-II Scheduling System

As mentioned at the beginning of this section, RESS-II is the second version of RESS. It has implemented both the reinforcement and the constraint decomposition approaches. The scheduling environment is described in chapter two.

This system consists of four parts, a reinforcement building part, a schedule construction part, a schedule refinement part and a rescheduling part. Each part may have one or more stages of scheduling. The reinforcement building part has two stages of scheduling. The first stage concentrates on the due date constraint's satisfaction to build a capacity plan, to show the conflict situations between resources and constraints. The second stage tries to expose the conflict situations between the due date constraint and work-in-process constraint, using the capacity plan as its reinforcement. These two schedules will act as an overall view of the forthcoming events.

The schedule construction part has only one stage of scheduling. It is the main scheduling process (so it is also called "main scheduling"). It uses the two reinforcement schedules from the last part to generate a schedule.

The schedule refinement part also has only one stage of scheduling. It refines the schedule constructed in order to satisfy as many local preference constraints as possible while also maintaining the previous stage constraint satisfaction result, using only the scheduling result from schedule construction part as the reinforcement.

The rescheduling part has only one stage of scheduling as well. Its function is to deal with reactive problem or schedule revision, using the capacity plan and the final schedule as reinforcements.

RESS-I is the first version of RESS. It also implemented the reinforcement approach. The difference between RESS-II and RESS-I is that RESS-I did not use a constraint decomposition approach. Consequently, the conflict situations between the constraints themselves are not tackled, and all the constraints are considered in a single schedule construction process. As the result, the schedules produced by RESS-I are not as good as those produced by RESS-II. RESS-I does not have a rescheduling part either.

1.5 Overview of the Thesis

The remainder of this thesis focuses on the issues surrounding the reinforcement reasoning approach used in job shop schedule construction. In chapter two, the scheduling environment is presented, in which we examine the nature and the complexity of the job shop scheduling problem: a variety of constraint satisfaction and resource allocation problems are described and past scheduling research both in management science and knowledge-based systems are reviewed.

This is followed in the next two chapters by further analysis, in which underlying problems are identified and new approaches to deal with them are introduced. Chapter

three first discusses the conflict situations between constraint satisfaction and resource availability, and then proposes the reinforcement scheduling approach. Chapter four first discusses the conflict situations between the constraints themselves, and then proposes the constraint decomposition approach. These two chapters contain most of the original ideas in this research.

Before going on to describe the RESS-II scheduling system, chapter five discusses the modeling of the scheduling domain, in which all the necessary scheduling concerns are represented.

The full account of the RESS-II system appears in chapter six and chapter seven. Chapter six focuses on schedule construction, which includes reinforcement building and detailed scheduling. Chapter seven describes work on monitoring and rescheduling.

After showing some experimental results in chapter eight, chapter nine makes a comparison between the approaches developed in this research and the existing ones.

Finally, along with the conclusion, chapter ten suggests the scope for future work.

1.6 Review of Related Work

Current AI planning research is just beginning to tackle problems in temporal and resource-restricted domains [8] [9]. The central idea is still no more than arranging the ordering relations by detecting and correcting interactions. Most systems implicitly assume unlimited availability of resources, and simply check their partial solutions against the actual resources available. In real world situations, activities may be constrained in their sequence and their execution times by the availability of resources as well as other constraints. Furthermore, resource allocation is not just the problem of checking the ordering solution against the resources available. More importantly it is an organizational problem of how to assign resources to actions to best satisfy various constraints. It is so acute that without this capability, any system will not have practical

applicability. As planning becomes more realistic, the demand for handling the resource restriction becomes an important problem.

Most work that has been done in this area is in knowledge-based systems. These systems tend to deal with the real world problems, and in most real world problems, resources and constraints often play the essential part. But, they are mostly based on simple expert or heuristic rules.

This section will only review related work in AI planning, while leaving past job shop scheduling research to the next chapter after the job shop scheduling environment has been discussed.

1.6.1 Resource Allocation and Constraint Satisfaction

O-Plan [8] [10] is an AI planner currently under construction at Edinburgh. It builds on earlier work from NOAH [6], NONLIN [7] and DEVISER [9], using a least-commitment approach, but presents a mechanism for handling temporal constraints and strictly consumable resources.

In O-Plan, a pair of numeric bounds are used to represent any uncertain numeric quantity. The temporal constraint of each planning network node, which represents an activity, is expressed by time windows which are specified by the earliest time and the latest time instance. Thus each activity has an earliest start time and latest start time, and an earliest finish time and a latest finish time. The lower bound of a time window represents the temporal constraint between the start of a plan and the current node. The upper bound represents the constraint between the finish of a plan and the current node. Computation of these bounds has to be done by finding the longest path in the plan state network. The successive refinement of the partial plan will narrow these time bounds.

Handling of a consumable resource is another important feature of O-Plan. Such a resource has an initial quantity and can be gradually depleted by the actions as it is consumed by the actions. In handling a resource, the time when it is consumed is not

important but the quantity is. Like the time window representing a temporal constraint, the quantity of this resource is also represented by a pair. In planning, each node has an attached resource usage window, and a resource propagation algorithm computes and maintains min-max pair information. There are four constraints in the resource window management. According to these constraints, the algorithm checks the resource pair at each plan node for mutual unsatisfiability of the constraints. After any bound is adjusted, a new check will be made on the modified resource bounds in consideration of the existing bounds. Supposing, the minimum overall usage is L (normally 0) and the maximum usage is U which is the initial quantity available. A plan will be valid if the total usage of the resource in the plan lies in the range [L,U].

Miller [11] recognized the problem associated with the least commitment planners of the NOAH kind (these planners produce plans in a partially ordered set of actions). It is fine for domains without constraints and resources, but will be inadequate for domains that involve deadlines, cost, efficiency concerns and resource. The reason for this is that the plans constructed by a least commitment approach cannot represent the quantitative interactions between the actions in the partially ordered plan network. It is also pointed out that like scheduling, planning often deals with temporal constraints and in many situations these temporal constraints are context-dependent which means that they may change according to different orderings (orderings here include both orderings and sequencings in our discussion. In fact in Miller's thesis they are unfortunately mixed up hence proving confusing). In his research, Miller proposed a planning paradigm in which such quantitative issues can be reasoned about and planned in detail. His approach is to explore the total orderings instead of partial orderings of the various plan actions to reach a result that will fullfil the quantitative constraints. He argued that unlike the NP-hard scheduling problem, many task scheduling problems have a reasonable number of orderings (or sequences). By simulating all the orderings, a required plan will be found. Even for some problems where their total orderings are impossible to track, there could be constraints in the realistic situation which make the total orderings possible.

When problems grow really large, reducing the search space becomes an essential task. To accomplish this, heuristic rules and a rating scheme are used to assess the partial schedules on their probable efficiency, and then to expand the most efficient looking schedules. Backtracking is also allowed when the predicted ordering leads to a failure.

The basic point in Miller's approach is to simulate the total orderings to find a good sequence of actions for the task given. But for most large tasks such as scheduling, heuristic rules alone are not sufficient, and global information is required.

1.6.2 Opportunistic Planning

MOLGEN built by Stefik [12] [13] is a constraint-directed system that assists molecular geneticists in planning/designing experiments. It was the first planner to highlight the use of constraints. In this system, the constraint posting technique was introduced.

Constraints in MOLGEN are represented as predicates, and they can be interpreted in three ways. Firstly, constraints are elimination rules, which means they constrain the set of allowable bindings, and if a potential selection does not satisfy the constraints, it will be eliminated. Secondly, constraints are partial descriptions and commitments. MOLGEN can formulate constraints on an object as its partial description without making specific selection. Thirdly, they are a communication medium, linking and expressing the interactions between different subproblems. The system also performs three operations on the constraints; constraint formulation, constraint propagation and constraint satisfaction. Constraint formulation is a process of adding new constraints in the design process, through which the system can proceed hierarchically into more detail. Constraint propagation is a process of creating new constraints from the old ones. It functions in MOLGEN as the communication between different subproblems. Further, as subproblems are normally under-constrained, such a process can further specify them by searching through different part of the problem to bring together the requirements. Constraint satisfaction is a process of finding values for the variables representing the constraints.

In control terms, instead of taking a fixed directional (or systematic) approach to solve a problem, the current focus of constraint satisfaction drives the MOLGEN planning process forwards. This approach integrates a least commitment strategy and a heuristic strategy.

However, the constraints in MOLGEN are in fact requirements of various experiments rather than constraints that restrict the planning result.

A cognitive model for an errand planning task was developed by Barbara and Fredrick Hayes-Roth [14]. It modelled a mixture of the goal-driven and current-situation-driven planning process observed in human planners (this process is termed opportunistic planning). They argued that people's planning activity is largely opportunistic rather than systematic. That is to say that at any point in the planning process, the current situation which includes both current decisions and the observations suggests many opportunities for plan development, and the planner's subsequent decisions may follow up these opportunities. Sometimes, the planning process follows an orderly path leading to a solution. This is described as a top-down process. But some other times, local observations and decisions can also influence the sequence of the orderly path by suggesting some opportunities so that the planner will follow a less systematic approach. This is a mixture of top-down and bottom-up processes. In planning terms, it can be described by stating that a decision at any level of planning specifying a plan action may influence the following decisions at higher levels or lower levels of abstraction.

The model assumes that the cognitive world consists of many specialists (different specialists may influence different levels of abstraction) and each of these specialists makes tentative decisions for incorporation into a tentative plan. This tentative plan is then evaluated against certain criteria. The whole plan process in this paradigm proceeds through a series of "cycles" during which various specialists execute their actions. At the beginning, it chooses a specialist invoked to generate a new decision which is recorded on the blackboard. The new decision will invoke additional specialists

and a new cycle begins. This process continues until a plan has passed the evaluation criteria.

The blackboard [15] [16] is used as the common data structure in this system. It allows decisions at arbitrary levels to be recorded, and this enables the specialists to interact and communicate.

1.6.3 Execution Monitoring and Replanning

After a plan has been constructed, it has to be executed. But, there are few domains where a planner can be 100% sure that each action will be executed according to the plan and will produce the desired effects. Typically, many unexpected interruptions will occur during execution. The old plan then needs to be updated to suit the new situation.

SIPE built by Wilkins [17] [18], which is also a NOAH like domain-independent planer, has been extended to perform the plan execution monitoring and replanning task. In SIPE, this process is accomplished with 4 steps. In the first step, it is input with information about the current unexpected situation in terms of what has been performed and what changes have occurred during the execution process. This can be done by some person or computer system monitoring the execution. In the second step, it determines the problems that the unexpected situation has caused in the old plan. There are two aspects to this: the first involves planning decisions that were based on the effects of the node that has gone wrong, and the second involves deductions about the state of the world that were based on those effects. For this purpose, SIPE has a problem recognizer which is capable of finding six kinds of problems, which are claimed to constitute the only things that can go wrong. In the third step, the system tries to fix the old plan, possibly by deleting part of it and inserting some newly created subplan. Instead of forgetting the plans from the problem node and planning again, SIPE makes changes to the old plan, keeping as much as possible of this plan. There are eight domain independent action operators which exist in SIPE to alter a plan. In the fourth step, it determines whether any changes made by the fixes introduced in step three will conflict

with the remaining parts of the old plan. Actually, step four is a part of step three, as only those fixes that are guaranteed to work are produced. Resource allocation and constraint satisfaction is not an issue in the system.

As mentioned previously, job shop scheduling is a highly dynamic problem. A considerable amount of the past research has been devoted to monitoring and rescheduling aspects. It deserves mention here.

Perhaps, the most important works are OPIS [19] reported by S.F Smith and SONIA [20] by Anne Collinot and Claude Le Pape. Both of these systems take the same approach, combining constraint propagation and consistency maintenance techniques with heuristics that map characteristics of current solution constraints to specific scheduling and rescheduling strategies (embodied in Knowledge sources). Their control framework draws from principles of standard blackboard style architectures [16].

These two systems basically consists of three main components, a schedule management module, a constraint propagation system, and a number of scheduling and rescheduling knowledge sources (KSs).

The schedule management module is responsible for evaluating the consequences of the scheduling decisions made and of unexpected events happening on the shop floor. It creates the initial set of temporal constraints according to the current situation. These constraints are passed to the propagation component of the system which derives new constraints from the existing ones. Having finished the propagation, a conflict analyser is used to examine all the constraints to detect the conflict situations. This process results in various proposals in order to solve the conflicts. Some very simple rules are employed for this purpose. According to these proposals, the required KSs are evoked to change the schedule to suit the current situation.

Planning and coordinating the scheduling actions to be taken in response to a given scheduling problem is carried out by a KS called "the manager". The information about the results of KS activity and schedule changes is conveyed to the manager via the

posting of control events. In turn, the manager responds by applying a set of event processing heuristics, which results in the formulation of new scheduling tasks. Thus the manager implements a reactive approach to control, continually revising its "scheduling plan" as the results of KS execution become known.

A purely reactive scheduling system reported by Elleby et al [21] works in a slightly different way by allowing human assistance in the decision making. It has also three major components, a constraint maintenance system, a reactive schedule generator and a request interpreter. The set of constraints defining the class of feasible schedules is stored in the constraint maintenance system, which is able to reason with both temporal and relational constraints. The reactive schedule generator constructs a schedule by trying to satisfy these constraints.

A schedule, once generated, can be judged by the human scheduler. Any criticism will be translated into constraints by the request interpreter for schedule revision. A work-in-process tracking system, which monitors the events that change the state of the plant, also provides input to the request interpreter. These events are then translated into constraints.

However, these systems do not have strategies which ensure that scheduling and rescheduling decisions made will necessarily produce good overall results. The problem of quick response may also be a problem as the constraint management system is time consuming (SONIA has introduced a number of control heuristics to constraint propagation system so that it runs more efficiently).

More detail of past research work into scheduling will be given in the next chapter.

1.7 Summary

This chapter gave a brief introduction to this thesis, in which resource and constraint problems in planning and scheduling were identified. Past work was also reviewed.

Along with them, the focus of this research has been pointed out, i.e. how to allocate limited resources in order to satisfy more constraints, and the ideas developed have been briefly discussed. Based on this chapter, from the next chapter onwards we will start to concentrate on the job shop scheduling problem and to propose approaches to deal with resource allocation and constraint satisfaction.

Chapter 2

Scheduling Environment and Past Research

Although this research is about the resource allocation and constraint satisfaction problems in AI planning and scheduling, the study carried out is based on job shop scheduling, which is an outstanding problem of resource allocation and constraint satisfaction. Chapter 1 has presented the problem and reviewed the past AI research in this area in general. This chapter will relate to the job shop scheduling task and introduce its problems.

The discussion is divided into two parts. The first part describes the shop environment, in which the scheduling task is defined and all the manufacturing terms are explained. The second part briefly reviews past scheduling research, which includes the work done both in management science and knowledge-based systems, but the emphasis is on knowledge-based scheduling systems.

2.1 Job Shop Scheduling

2.1.1 Scheduling Environment

A job shop is characterized by the need to process a large number of products, which are specified by customer orders or forecasts, on a relatively small number of machines. The number of orders can be more than 200 and the number of machines can be more than 50 (machines are grouped into work centres). Each order requires a product which is manufactured with a different number of separate operations. The operations have to be performed in a specified sequence. The total capacities of resources available are restricted.

Scheduling can be defined as: (1) prescribing when and where each operation necessary to the manufacturing of a product is to be performed, or (2) establishing the times at which to begin and/or complete each event or operation on a machine. In production, it is a management function that is concerned with planning the sequence in which different operations should be performed on machines and also with planning the starting and/or finishing times for these operations. The inputs to scheduling are the orders and the shop environment which includes machines, tools and process routings. The outputs are schedules for machines, tools and operations (or orders).

Now I will explain some manufacturing terms. Note that this is limited only to the ones that are necessary for the scheduling task.

- Job shop

  The job shop is the place where all the actual manufacturing activities take place, i.e. it is the place all the management decision and production plans or schedules are realised. It can be described both from a physical layout perspective and an organizational perspective. From a physical layout perspective, a job shop contains a number of physical machines on which

manufacturing operations are performed. From an organizational perspective, the production process and its physical environment on the shop floor are described, such as physically identical machines being grouped into work centres. A job shop is also an element in a complete manufacturing system.

- Resources

Resources in a job shop include all the items that enable the production process to be carried out, such as machines, tools, fixtures, NC tapes, jigs, etc. (For simplicity, from now on, we will use the term "tool" to represent all such resources other than machines.)

- Manufacturing routing

Manufacturing routing is the function that determines the path that each internally manufactured order will take through the production plant. It prescribes a sequence of operations that will transform the input materials into the desired end products. Its information comes from the process planning methods, such as time and motion study, which is another stage of manufacturing management activity. There are two types of data contained in a routing:

(1) Technical data. This data instructs the operator how to perform an operation — the cutting speed, feed rate, depth of cut, inspection, and so on — but generally does not participate in the scheduling task.

(2) Management data. This data specifies the sequence of operations and the information required to determine the work centre load (running time and setup time). It also contains other information, such as tools that will be used and alternative work centres. This is the set of data used by the scheduling function. Consequently, only this set of data is stored in the routing file.

- Orders

Orders are the requests for particular products which can be manufactured in the plant. Different kinds of orders exist:

(1) customer orders which are directly required by the customers
(2) forecast orders which are the company's forecasts according to market
(3) stock orders which are stored in stock for any possible requests

Manufacturing activities start and end with the orders, and scheduling is based on orders. They initiate all the subsequent activities of the manufacturing process, which are finally terminated by the delivery of goods. Each order may contain a lot of information, but as far as the scheduling is concerned, it basically has the information about order number, part number, quantity required, due date, start time and importance.

2.1.2 Scheduling Constraints

Job shop scheduling is the assignment of time bounds to specific manufacturing operations with respect to the finite set of resources (such as machines and tools). In other words, it is a problem of proper allocation of these resources such that the manufacturing can be carried out in a timely and cost effective fashion. It is a difficult task. Much of the difficulty comes from the need to allocate limited resources to attend to a large and diverse set of objectives, requirements, and preferences. The conflicts between these scheduling influences are often not clear. Hence, a good schedule must reflect a satisfactory compromise among these competing influences.

The schedule influences or constraints can be partitioned into three classes: (1) scheduling organizational goals or objective constraints, (2) local constraints, (3) technical constraints.

(1) Organizational goals or objective constraints

The organizational goals are aimed at reducing the cost as much as possible. They are the primary concerns or objectives within the factory, against which the factory production is planned and the shop activities are scheduled. Decisions related to these goals and constraints are made on the basis of current and future costs. For example, failing to meet customer delivery due date may result in a lost customer. Examples of such goals and constraints include:

Meeting due dates
Minimizing work-in-process time
Maximizing resource utilization
Maintaining shop stability

— Meeting due date: Due date is the time when a part should be finished. This means that the product required by an order has to be ready at this time for shipping to a customer, or must be ready to be brought for assembly. It is the primary concern in any factory. Failing to meet a due date may result in customer dissatisfaction or losing customers. So, the aim should be to plan the sequence of work so that production can be organised towards the aim of completion of all products by due dates. The due date for each order is an external datum from a management level and is dictated in the order.

— Minimizing work-in-process time: Work-in-process is the time period over which a part is on the shop floor. The investment of raw material will not be recovered until order delivery, and the cost of handling the part on the shop floor will increase as the work-in-process time goes longer, so to prevent capital being tied down in the manufacturing process the work-in-process time should be reduced to a minimum.

— Maximizing resource utilization: The objective of minimum idle time on machines, implies the maximum utilisation of the plant. How this constraint can be achieved depends on how the loads on machines are balanced. In any manufacturing situation, there are normally several machines that will be overloaded during the time period concerned.

There is excess capacity on most of the remainder of the plant. To achieve maximum resource utilisation, particular attention should be paid to those heavily loaded machines. If, instead, attempts are made to minimize machine idle time on the remaining machines, the resulting effect will be to increase the investment in work-in-process and in stocks in the inventory.

— Maintaining shop stability: The concern here is to minimise the amount of disruption to scheduled operations caused by revisions to the existing schedule.

(2) Local constraints

In addition to the objectives which are designated to maximize profits, preference constraints express the preferred choices among alternatives. Their satisfaction can improve a schedule. For example, saving set-up time will reduce the load time that an operation spends on a machine. This class of constraints includes: set-up time, resource preference and job importance and urgency.

— Set-up: In most manufacturing operations there will be a set-up time during which the operator mounts the holding fixtures on the machine, places the part in the fixture, places the cutting or forming tools in the machine spindle, puts the NC tape in place, establishes axis datum points, and checks out the set-up. The time taken and the cost of setting up the machine depend on the nature of each operation. If the set up is the same between two successive operations, then a set-up time period could be saved, which will in turn reduce the manufacturing time or increase machine capacity. It is a constraint that depends on the previous operation.

— Machine preference: Although the machines in each work centre may be physically identical, they will have some differences because of factors, such as tolerance, cost and quality. The skill of its operators could also be different. However, factors like cost and quality are difficult to assess, so it is the shop supervisor's own judgment that a particular machine should be used to process certain operations.

— Job importance and urgency: Not all the orders which come to the shop are equally important, and not all of them are equally urgent. Some orders are more important than others for various reasons, and some orders are more urgent than others, because of their short due dates for instance. Job importance is a management assigned value to express the relative importance of an order among the other orders according to various factors, while job urgency is a value which can be calculated.

(3) Technical constraints

These constraints differ from the first and second categories in that those earlier constraints can be relaxed during the scheduling process while technical constraints are fixed. For example, the non-availability of certain resources during the manufacturing means that an operation cannot be performed.

— Resource requirements: There are resource requirements, such as machines and tools associated with an operation during the manufacturing process. In other words, the operations have to be performed on machines with the presence of certain tools. The allocation of these resources will finally decide the satisfaction of both objective constraints and local constraints.

— Precedence restriction: Operations in an order cannot be performed randomly. As specified in the manufacturing process routing of a part (or a product), there is a precedence restriction associated with the operations. This constraint is laid down by the production engineer to restrict the routes of the product going through the factory. It is actually an ordering constraint (this has been mentioned in chapter 1) on an operation that must be performed before or after other operations.

2.1.3 The Dynamics of a Job Shop

A working plant is a dynamic environment, subject to many changes and unplanned interruptions, which may in turn lead to the accumulation of unrequired stock, missed

due dates, etc. A schedule is simply a forecast and changes will be forced on this forecast by these interruptions. Usually, the life of a schedule or plan is no longer than a day or so. Typically, the unexpected interruptions may include:

Machine break down
Non-availability of resources
Operator absenteeism
Customer orders being added or deleted,
Quantities being altered
Delivery dates being changed
Late or early finish of the operations

With so many interruptions, the capability of any scheduling system is also measured by how well it can respond to change while maintaining shop stability (keep as much of the existing schedule as possible).

2.2 Research in Management Science

Past research on scheduling suggests that there is seemingly no simple way of dealing with the problem. Some people think of it as a science, and many papers have been published on the theories of job shop scheduling [22] [23] [24]. On the other hand, some people think of it as an art and believe that the skill and experience of the foreman can effectively load the shop. However, as the number of the jobs and the number of machines grow, the scheduling task becomes so large that human schedulers cannot handle it effectively. In fact, scheduling is somewhere between these two extremes [24]. During the past three decades, a great deal of effort has been devoted to it, and many approaches have been developed. They can be categorized into five general groups: analytical, iterative, heuristic, Gantt charts and computer programs [22] [24] [25].

- Analytical approach

  Several attempts have been made to structure the scheduling problem into a formal mathematical model. For example, the determination of an optimal job

shop schedule can be formulated as an integer programming problem. However, scheduling is a combinatorial problem or NP-hard problem, so the computational demands are often severe. In addition, a large number of assumptions are made which rarely hold true in real situations. The overall aim has been accomplished only for trivial problems involving at most several machines.

- Iterative approach

In this approach, all possible combinations of task sequences at each facility are tried and the best combination is chosen. Conceptually, this approach is good, but it is not practical because of the tremendous amount of time and computational effort required. Although various ways can be used to cut the number of possible schedules, such as restricting the possible schedules to active or nondelay ones [23] and by using the branch and bound method [23], the resulting number of possible schedules is still too large to be practically enumerated.

- Heuristic approach

Because of the computational problems with the analytical and iterative approaches, it is necessary to consider heuristic methods. Over the past years various heuristic methods have been developed, using decision rules and simulation models. In such approaches, the computational time can be significantly reduced, but they are not easily adapted to different situations. The most commonly used heuristic method employs the logical decision rules (or priority dispatching rules). These rules may include:

EDD (Earliest Due Date): Select the operation whose order has the earliest due date.

FCFS (First Come First Served): Select the operation that enter the waiting list of a machine earliest.

LWKR (Least Work Remaining): Select the operation whose order has the least work remaining to be processed.

MOPNR (Most Operations Remaining): Select the operation that has the most operations remaining in the same order.

MWKR (Most Work Remaining): Select the operation whose order has the most work remaining to be processed.

RANDOM (Random): Select the operation at random

SLACK (Smallest Ratio of Slack to Number of Remaining Operations): Select the operation that has the smallest ratio of slack time to number of remaining operations in the same order.

SPT (Shortest Processing Time): Select the operation with the shortest processing time.

These are a few simple rules for different situations. There are also some other very complicated rules which are intended to cover more aspects of the job shop scheduling environment.

It seems that it is not difficult to design a plausible dispatching rule. In some situations the rationale for using a particular rule may be that it helps relieve congestion in the shop, while in other instances the motivating factor may be a need to meet order due dates. A large amount of simulation work has been done to examine these rules, but many assumptions are made and simple working environments are used. They do not generally consider resources other than machines.

Use of a single rule is only one of the ways to construct a schedule. Sampling is another. Although the enumerative approach is impractical for the solution of large problems, the construction of single schedules by priority dispatching rules may still be a considerably easy task. It is reasonable to think in terms of repeating the generating process many times with some kind of change to the

rule each time or using different rules. In this way, the best of several schedules can be selected.

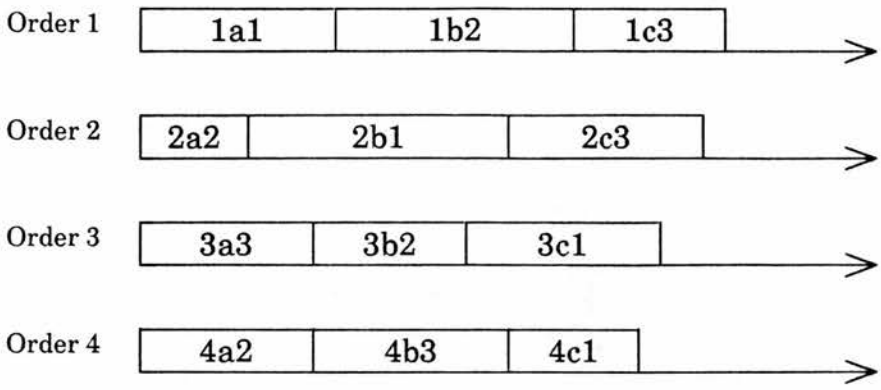
- Gantt Chart

Of the many techniques employed to support the scheduling, the oldest is the Gantt chart. This chart is relatively inexpensive and can be easily and effectively used for scheduling purposes. It provides the planner with critical insight into the inter-relationship among the specific activities within an entire process. Even the most complicated production plans can be presented clearly on a chart, which makes it an excellent communication device; understanding and interpreting a chart is relatively easy and, what is more, it can be done accurately and consistently by all informed observers.

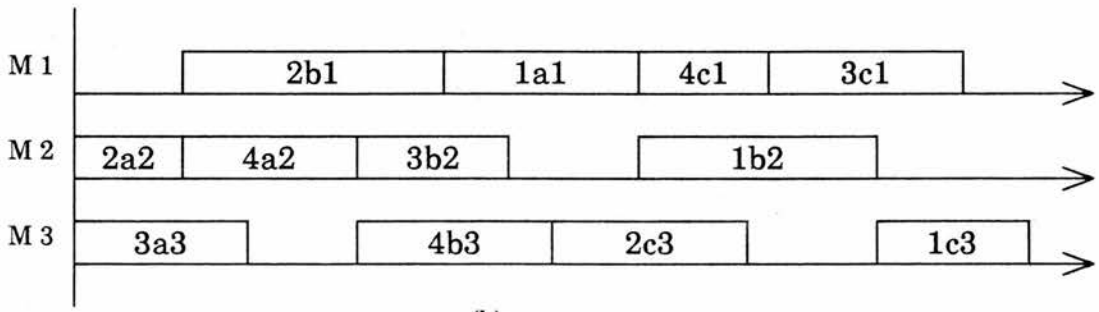
Basically, a Gantt chart is a device for depicting a work plan in terms of time and for showing the progress of the work in relation to the plan. This chart may be adapted to any type of manufacturing process and any activity within that process. There are three basic varieties of Gantt charts:

(1) Schedule charts
(2) Record charts
(3) Programme process charts

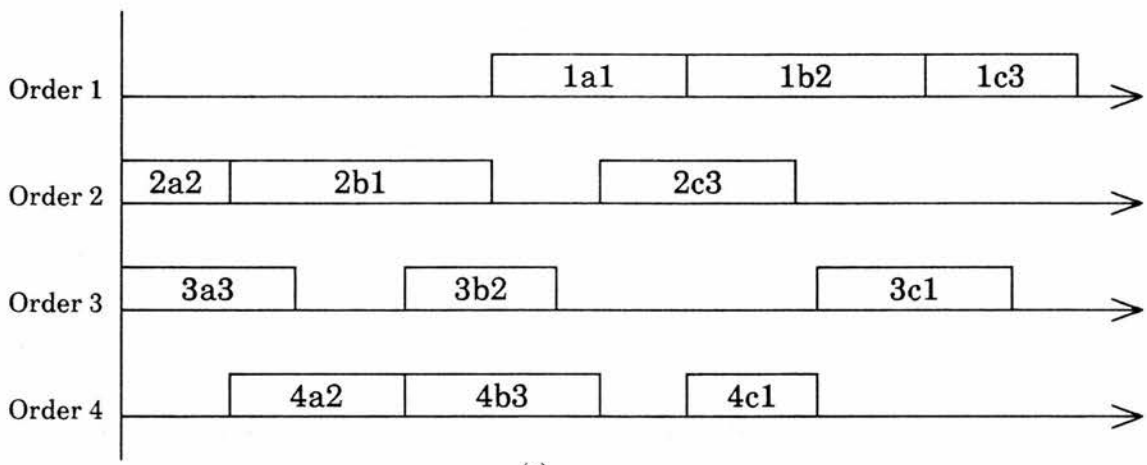
Here we will only briefly describe the schedule chart. The other two are similar. The purpose of a schedule chart is to show the operation assignments that have been made to a plant. Operations are typically measured in units of time. Therefore, what is recorded is the amount of time that the operations will take, and not the particular operation being performed or the nature of the operation. Figure 2.1 illustrates an example. Each operation in a job (or an order) is described with a triplet (**i**, **j**, **k**) in order to denote the fact that operation **j** of job **i** requires machine **k**. In Figure 2.1, (a) shows a set of given orders and their operations. (b) shows a feasible machine schedule with Gannt Chart, (c) shows the operation schedule in each order.



(a)



(b)



(c)

Figure 2.1 Gantt Chart

(a) Order routing description (b) Machine schedule Gantt Chart
(c) Order schedule

- Computer systems

As the computer is being used in job shops, many computer systems have been built to perform the job shop scheduling task. However, since even modest sized problems have an enormous number of possible schedules, an optimal solution is usually impossible to obtain by inspection or by complete enumeration of all possibilities. The existing systems are mostly heuristic programs, which seek to produce one feasible schedule.

Ferguson and Jones [26] developed a job shop scheduling system in 1969. The shop example they used was relatively small, consisting of just six machines. By choosing the proper values of a number of operational parameters, the person doing the scheduling had the option of making all decisions himself or of making most of them by computer.

PRODUCE, a system built by Heuristima seems to be more powerful. Utilizing sequence programming, PRODUCE rapidly converges on an initial shop schedule. If this is not acceptable, the user may interact with the system to make changes and then ask the system to give a revised trial solution. It also supplies the user with all the operating documentation he needs to actually run his shop. Unfortunately, though, there is no published material available on it and it has not being extensively promoted.

Perhaps the most widely exposed heuristic program is SPAR-I, developed by Wiest [27]. Although it is a system for scheduling large projects and not for job shop scheduling, the underlying ideas are the same. SPAR-I is based on a parallel strategy for its schedule generation. At the centre of the parallel scheduling in SPAR-I is a priority scheme based on urgency. At any stage in the construction of a schedule, the activities whose predecessors have been complete are considered for scheduling in order of their priorities. As the algorithm moves forward in time, the priority of each schedulable activity is revised, so that a dynamic priority scheme results.

Other systems dealing with job shop scheduling also exist. They more or less fall into the same category. These earlier systems are mostly simulation programs which attempt to assist human schedulers. They usually only meet one or other scheduling objective, most probably the due date objective and attempt to produce a schedule which gets as many jobs out on time as possible, while neglecting other important objectives. Some of them just use simple dispatching rules to arrive at a schedule. Later, with the advance of artificial intelligence, some expert scheduling systems were built by various people. ISIS [28] built by Fox is representative of the most sophisticated.

2.3 Knowledge-Based Scheduling Systems

2.3.1 A Brief Review of Expert Systems Structure

An expert system is a computer system that can solve problems using human expertise and knowledge of the system environment. In order to build an expert system, one has to consider the following issues:

— Knowledge representation
— Inference engine
— Knowledge acquisition

These three aspects will be briefly discussed below.

Knowledge representation:

There are a number of ways to represent knowledge in expert systems. Three main formalisms have found favour with expert system designer: production rules, structured objects (or frames) and predicate logic. Perhaps the most widely used formalism is the use of production rules.

(1) Production rules

Production rules are sometimes called 'condition-action rules' or 'situation-action rules'. Their principal use is in the encoding of empirical associations between patterns of data presented to the system and actions that the system should perform as a consequence. A production rule may have one of the two forms:

If <conditions>
Then <conclusions>
or
If <conditions>
Then <actions>

The rule can be further classified into two categories: application specific rules (or object level rules) and meta-rules. The application specific rule are used to execute actions or to make change to the situations, while the meta-rules are used to control and reason about the application specific rules.

(2) Frame representation

Frames are a way of grouping information in terms of a record of 'slots' and 'fillers'. They are used to represent descriptive knowledge and facts. A frame can be defined as a structured piece of knowledge where slots are corresponding to attributes related to the object it represents. Consider the following simple frame, which is really no more than a record structure.

Frame: Drilling-operation
Is-a: Operation
Work-centre: work-centre-25
Tool: Tool-13
Running-time: 25min
Set-up-time: 60min
Next-operation: Milling operation

This information could represent a stage of manufacturing a particular component and it means that this drilling operation requires work centre 25 with tool 13. The running time and set up time are 25 and 60 minutes, respectively. Its next operation for the component is a milling operation. Many systems use frame representation for their knowledge representation and these systems are called frame based systems [29].

(3) Predicate Logic

This is a formal language, namely a calculus with syntactic rules of deduction. An argument consists of a number of statements or propositions from which some other propositions or statements result. The initial statements are called the premises, while the latter is called conclusion. These statements are described by a sequence of symbols. The meaning, value or outcome associated with an expression in the language depends solely on its external form, and not on any extraneous associations or ideas that might be attached to the symbols in the mind of someone reading them. This representation scheme is normally used in theorem proving.

For example, to express the fact that "if X is a human being, then X must be a man or a woman", the statement could be like this;

$$\text{Human-being}(X) \longrightarrow \text{Man}(X) \vee \text{Woman}(X)$$

Inference engine

The inference engine is the part of an expert system that manipulates the rules and the frames, etc. or uses knowledge to work with problems. It normally contains an interpreter and a control strategy. For instance, the interpreter for a set of production rules can be described in terms of the 'recognize-act cycle', and it basically consists of three steps.

(1) Pattern match: Match the calling patterns of the rules against the facts in the data base.

(2) Fire the right rule: If there is more than one rule that can be fired, then decide which one to fire, i.e., employ a firing strategy

(3) Apply rule: Add a new fact to the data base or delete a fact in the data base according to rule application. This is often held as an "add-list" and "delete-list".

The strategy in the expert system can be compared with the algorithm in any conventional computer program. There are global strategies and local strategies. The global control strategy tends to be domain independent, in that the strategy employed does not use domain knowledge to any significant extent. The local control strategy is normally domain-dependent, in that special rules and procedures are used to reason about the control.

At the global control level, the reasoning process can be driven forwards or backwards, hence are called forward reasoning and backward reasoning systems. For instance, in rule based systems, forward reasoning works from the condition of the rules that we know to be true, toward the conclusions which the facts allow us to establish, by matching data in the data base with the left-hand sides of rules. Backward reasoning works in the reverse order, in that the conclusion is the facts we know and the reasoning is toward finding the facts in the world data base which enable these conclusions to be true.

At the local level, in rule-based systems, each production rule can be self-contained, which means it does not have any direct connection with other rules, or it will never call other rules, and all necessary context for rule activation is provided by the premises. Sometimes, the activation of one rule may facilitate the use of another rule. This does not mean it calls the other rule, rather one rule is activated via the changes another rule makes to the data base.

In some systems, meta-rules are used to control the sequence of the application of object level rules. Meta rules are distinguished from ordinary application specific or object-level rules in that their role is to direct the reasoning required to solve the problem, rather than actually perform the reasoning.

In state space search, there are basically two control strategies at a local level, backtracking and heuristic search. Operators are often developed and used to change the states in the search process. In a typical situation, there is more than one state to be considered. If the choice chosen is not correct, backtracking is performed. However, this can be computationally expensive, especially in situations where many states are involved. So, most of the systems use heuristic knowledge to cut the search space. For example, most planning or scheduling systems use constraint-directed search, where constraints may involve time, resources, deadline, costs, etc. When this search method is applied, more information related to the environment can be used in the reasoning processes of the expert system.

Knowledge acquisition

This is a process to acquire knowledge from different sources for expert systems. For example, in a scheduling domain, there could be four sources from which the necessary knowledge can be gained,

— domain experts,
— global data base in manufacturing system,
— mathematical models,
— simulation programs.

Knowledge acquisition is normally a time consuming process.

2.3.2 Knowledge-Based Scheduling Systems

In the recent past many knowledge-based scheduling systems have been built by various people, using various techniques [30]. Some of them are rule-based systems, and

others are based on frame representations. Some of them only use heuristic rules to construct a schedule but others conduct a constraint-directed state space search. Some of them are only for shop loading or daily scheduling, and some others only build a complete schedule without worrying about the shop floor monitoring. Some perform both functions.

ISIS built by Fox [28] [31] is a constraint-directed reasoning system for the scheduling of a factory job-shop. The main feature is that it formalizes various scheduling influences as constraints on the system's knowledge base and uses these constraints to guide the search and heuristically generates the schedules. In each scheduling cycle, it first selects an order to be scheduled according to the priority rules and then proceeds through a level of analysis of existing schedules, a level of constraint-directed search and a level of detailed assignment of resources and time intervals for each operation in the order.

The system is based on SRL (Schema Representation Language), a frame-based knowledge representation language. It provides ISIS with the capability of modeling a plant at all levels of detail: from the physical machine description, to process descriptions, to organizational structures. For example, an operation can be described with the following:

```
( Operation
    Next Operation
    Previous Operation
    Machine
    Operator
    Duration
 )
```

The constraints are also represented with SRL. The constraints represent the schedule objectives, resources and preferences, etc.

The four level hierarchical structure in ISIS is as following:

Level 1 Order selection: this level selects an order to be scheduled according to a set of priority rules.

Level 2 Capacity analysis: according to the plant capacity, this level generates a time window for each operation in the order, specifying the earliest start time and latest finish time.

Level 3 Scheduling: this level performs a constraint-directed search among alternatives, with all the details considered to arrive at a schedule for the order.

Level 4 Reservation selection. This level reserves the resources this order needs.

The most important part of ISIS is level three, which uses constraint-directed search. In scheduling the order selected at this level, the search can be either forward from the order's requested start date or backward from the order's due date according to the nature of the order. The search space for scheduling each operation is composed of states which represent partial schedules, and the method of search is beam search. After any given operator is applied, the system rates the newly generated states by relevant constraints to choose the best to extend. Once a set of schedules has been generated (all the operations in the order have been scheduled), a post-search analysis is performed to examine if one is acceptable and whether rescheduling is needed. The outcome of scheduling is a routing for the order along with the assignment of time bounds to the resources required to produce it.

However, decision making at this scheduling (the third) level is only based on local information and what has been done in the past. The situation is thus: after an order has be selected at the first level, a capacity analysis is performed. This analysis will first check to see the schedules of the previous orders or the current scheduling situation (or resource reservation) on each machine. This capacity analysis detail is combined with detail of the durations of the operations to provide information indicating how each operation of this order may be scheduled. The third level and the fourth level will use this information in their decision making and resource reservation. However, such a capacity analysis does not consider what may happen in the future and what influence current decisions may have on the following schedules.

A large amount of the work done by ISIS actually involves the extraction and organization of constraints that are created specifically for the problem being worked on. In such an approach with this searching framework, scheduling relies only on

order-based problem decomposition. ISIS has not dealt much with resource allocation problems. The current OPIS system [32] [33], which is a direct descendant of ISIS, attempts to make some progress by concentrating more on the heavily loaded work centre(s). It adds one more perspective to the scheduling architecture, namely resource-based decomposition. As a result, it has two schedulers: one is an order scheduler which is the same as that in ISIS and the other is a resource scheduler. What this system does is to analyse the resource capacity to find the bottleneck machines and schedule the operations on these machines first with its resource scheduler. Then, it spreads the scheduling to the other machines using the order based scheduler. The decision making in resource scheduling is based on heuristic rules, and in the order scheduling it is the same as in the ISIS system.

A very different system called PATRIARCH for solving FMS problems including project planning and scheduling was built by Morton et al [34]. This system has a planning or scheduling model called MRP-STAR. The decisions in this model are based on a costing methodology, which is decided by considering the cost of scheduling, tardiness cost, inventory cost, cost of using work centre and so on. All these costs are dynamically priced. However, it is essentially a rule based system but based on different criteria for assigning priorities to the competing tasks.

The PATRIARCH costing methodology approach is to balance scheduling costs versus tardiness costs. The scheduling cost is taken as the avoidable direct costs and material costs at processing, plus the implicit cost of using the work centre, less immediate incremental revenues received. The tardiness cost is a function of lateness, and includes any one-time penalty, opportunity cost of lost revenue, etc.

The scheduling decision is made according to the costing methodology. The total cost of delaying the processing of a job on a machine is tardiness costs less the interest on the delayed scheduling. This cost is divided by the costs of delaying scheduling. The result is a rate of return on this job going first. When the work centre becomes available, dispatch the job in the queue with the highest rate of return. If all rates are negative, then schedule no job.

Before the job can be put in the queue for scheduling decision making, it has to be released from the list of available jobs. To decide which jobs should be released, the system calculates the rate of return of these jobs the same way as those jobs in work centre queues. Whenever the rate of return is above 0.0, it may be released to its first queue.

PATRIARCH also considers the dynamic issue of the shop, and it has several modes for dynamic revision of the plan according to the current shop situation.

(1) Dispatch mode. It handles minor problems and assumes that the priorities are still not changed. After considering all the fixed and dynamic constraints as they become known, the system still uses those priorities to choose the next job.

(2) Reactive mode. This mode works almost the same way as the dispatch mode, except that one or more job priorities will have changed too much for the assumption that the priorities are still maintained. The revised priorities are calculated, and then it goes into the dispatching mode. If the whole schedule is likely to be affected, the next (the replan) mode will be called.

(3) The replan mode. This mode works exactly the same as the original planning or scheduling.

An expert system called SOJA which is a daily workshop scheduling system was built by Lepape et al, [35]. This system focuses attention on the scheduling for the next day. It is actually a loading program which loads the jobs that are available to each machine according to the current shop situation. During the loading it considers some practical constraints which are embodied as expert rules. The system starts the scheduling with the building of a selection graph which take the current loading on a machine as the root node and extends this node by adding new jobs which are associated with some constraints. After adding all the jobs on to the graph, the system chooses an operation by going through some expert rules and comparing the weight of each job on its constraint satisfaction.

Bruno et al [36] developed an expert scheduling system for scheduling parts in a flexible machining environment. The parts are grouped into batches. Each batch contains 100 to 200 parts. The system uses a priority rule to decide the release time of a batch. The rule it uses is as the following:

(Remaining machining time / (due-date – release-time))

This decides the priority of each batch. But apart from this priority rule, the other constraints considered include:

(1) capacity of the queue,
(2) fixture or tool vacancy, and
(3) machine maintenance periods.

If the above constraints are satisfied, the batch with the highest priority is released to be manufactured.

The scheduling structure of the system consists of two modules:

(1) The scheduler. It is a expert system containing a set of rules for schedule generation.
(2) A load evaluation module. It is a simulation system for schedule evaluation.

Erschler and Esquirol (1986) [37] presented a job shop scheduling system, MASCOT, which uses a constraint-based analysis (CBA). The system makes resources (machines) constantly available and tries to finish jobs before the due dates. However, it only includes the machine resource and the operations in its scheduling. Start times of operations and constraints that use common resources are considered as important aspects of the problem. The CBA approach generates a sequence relationship among operations. The rules used are, consequently, of only two types:

(1) time updating rules
(2) sequencing rule for precedence among operations

Subramanyam and Askin (1986) [38] discussed an approach for scheduling a FMS (flexible manufacturing system), on a daily basis of two shifts to meet the weekly production requirement. The FMS is described by three object statuses:

(1) system status
(2) machine status
(3) job status.

The system status and machine status describe the loading conditions of the shop and the machines, heavily loaded, moderately loaded and under loaded. Similarly, the job status expresses the jobs in the queue of their urgency, critically late, moderately late and normal. Systematically, a three level hierarchical structure is formed. A set of expert rules are used to evaluate these three level or three object status to make decisions. The rules used are mainly acquired from job shop simulation. They suggest that the combined knowledge acquisition from domain expert and simulation is the appropriate way for expert planning and scheduling.

2.4 Summary

Scheduling is a complicated problem of both resource allocation and constraint satisfaction. The approaches developed in the past in management science have had only limited success. Most existing knowledge based scheduling systems are only capable of incorporating a small fraction of scheduling knowledge. As a result, the schedules produced bear little resemblance to the actual state of a factory and can only be used to provide a high level guideline for human schedulers. The ones that do have extensive coverage in the scheduling domain (like ISIS) only depend for scheduling on local information and what has been done in the past without considering what may happen in the future. Such a process is blind and its earlier decisions may create many problems later which otherwise would not exist. It is no better than using simple dispatching rules. In fact, the underlying scheduling problems have not yet been understood to a sufficient degree. In the next two chapters, we will discuss these issues further.

Chapter 3

A Reinforcement Approach to Scheduling

In the last chapter, along with describing the job shop environment and reviewing past scheduling research, the scheduling task was defined and general scheduling problems were identified. This and the next chapter will go into detail to further discuss these problems and propose approaches to deal with them.

To start this chapter, we first present two scheduling methods by which a schedule is constructed. Then, we analyse the scheduling process, within an AI framework, in one of the methods (the parallel method), during which we will introduce the detailed scheduling problems.

In such a scheduling process, two levels of reasoning activities can be identified at a scheduling state, a choice generation level and a decision making level (or choice making level). Choice generation is the logical aspect, satisfying ordering constraints (the precedence) and it is not difficult in the scheduling domain. However, the second level is actually the resource constrained decision making which is essential to scheduling tasks. Discussion of the problems at this level highlights the conflict situations between constraint satisfaction and resource availability. It is suggested that

the key to successful scheduling lies in having a global view. To deal with the problem, "reinforcement scheduling" is proposed.

3.1 The Scheduling Methods

In order to analyse the scheduling process and discuss the detailed scheduling problems, we must first know how a schedule is constructed and how the process proceeds. This section presents the scheduling methods.

There are many ways that can be employed to perform a scheduling task. But basically, there are two main methods: single-order scheduling and parallel scheduling. They are similar to the series method and parallel method in project scheduling [39].

The single-order (or order-based) method schedules one order after another. We know a product required by a particular order normally has several manufacturing operations which are performed on different work centres. In each scheduling cycle or at a scheduling state, this method first chooses a particular order according to some criteria from a set of pending orders. Then, it schedules the operations in this order one by one onto their machines until all of them have been finished. In the next scheduling cycle, another order is selected and the scheduling is performed in the same way. Some systems have used this method for their schedule construction, ISIS [28] and OPIS [32], for instance being the best known.

The parallel strategy schedules in a very different way. In this method, several operations from different orders are considered at once. At each scheduling state (or point in time when a machine becomes idle or ready to perform a new operation) during the construction of a schedule, there exists a set of schedulable operations whose predecessors have been completed (i.e. their precedence constraints have been satisfied). From this schedulable set of operations, a preferred one is chosen to be scheduled next according to constraint satisfaction and resource capacities. At some future state (or point in time), a new set of schedulable operations is encountered and the schedule is

constructed in the same way. The schedules, in this method, are created by proceeding chronologically forward. Normally, parallel scheduling is used in monitoring and rescheduling (or short term scheduling and dispatching) [23] [25].

To illustrate how these two scheduling methods work, let us suppose that there are two orders that need to be scheduled on four work centres. Work centre 1 has two machines (M1a and M1b) and work centres 2, 3 and 4 only have one machine each. Both orders have four operations. Order 1 has operations Op1a1, Op1b3, Op1c2 and Op1d4, and Order 2 has operations Op2a1, Op2b2, Op2c3 and Op2d4. An operation is described with a triplet (**i**, **j**, **k**) in order to denote the fact that operation **j** of order **i** requires work centre **k**. In the diagram, it is represented as a line of certain length and this length expresses the running time of this operation.

Firstly, the single-order scheduling method is used to schedule these two orders. The process starts by choosing a particular order: supposing for reason of priority, Order 1 is selected to be scheduled now. With this choice, the system begins to schedule every single operation of this order. Operation Op1a1 is the first operation and is taken first to be scheduled. In this case, at this point in time it can be scheduled as early as possible, i.e. time 0. This occupies the first machine M1a of the first work centre for a period of time. Operation Op1b3, which is the second operation of this order and requires M3 (from the triple), cannot be started until the first operation has been finished. With operation Op1b3 scheduled on M3, the method proceeds to the next operation (Op1c2) of the order and schedules it as early as possible. This operation scheduling process continues until eventually all the operations in the order have been scheduled or a complete schedule has been constructed for this order. Then, the second order is selected. All its operation will be scheduled in the same way. The final result is shown in Figure 3.1. Remember that work centre 1 has 2 machines.

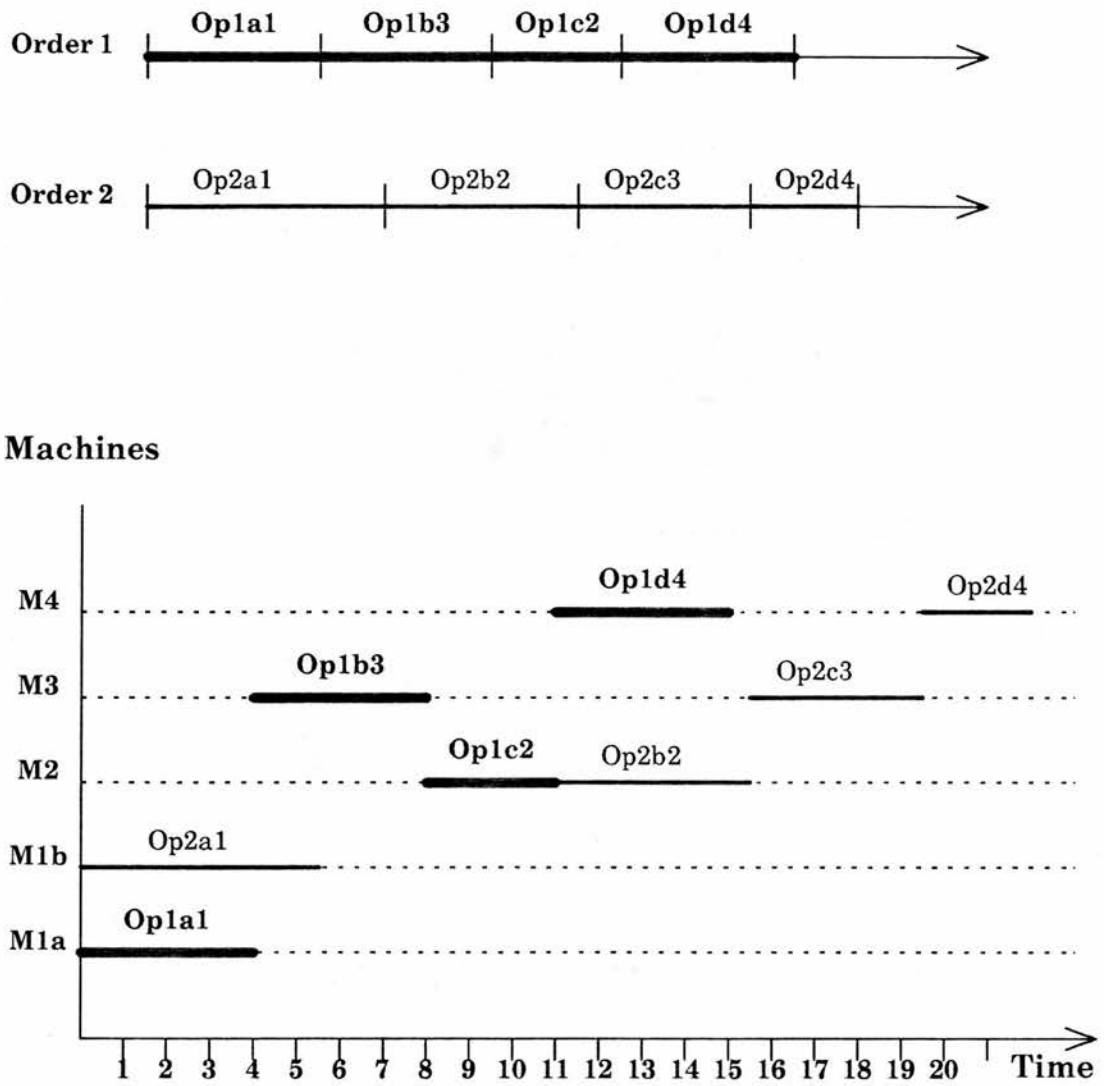


Figure 3.1 A schedule constructed with the use of the single-order scheduling method

Alternatively, the parallel scheduling method is employed to schedule the same orders on the same work centres. In this method, the first task is to set up a scheduling state, and hence to decide on which machine to schedule next. The machine whose last operation finished the earliest or whose idle time started the earliest will be chosen. This machine and its idle start time sets up a scheduling state. For example, in Figure 3.2, the current scheduling state is state M. Operation Op2a1 is chosen to be scheduled on machine M1a. The next schedule state will be state N. The reason for choosing the scheduling state in this way is to ensure that no operation is left unconsidered at any

scheduling state when it deserves consideration. For instance, if the system chooses state Q to be the current scheduling state instead of state M, and if the operation Op2a1's next operation Op2b2 can be scheduled on M2a, then Op2b2 will be missed from being considered as a candidate operation at this schedule state since its last operation has not been scheduled.

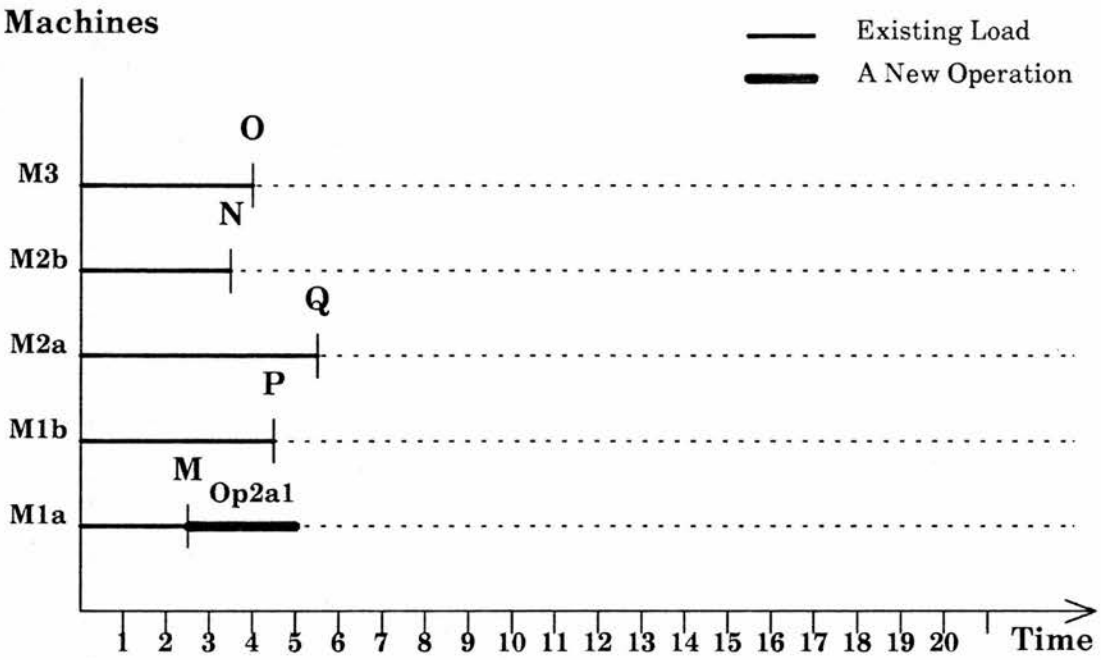


Figure 3.2 A partial schedule and the schedule states

According to the discussion above, to set up a scheduling state is to check each machine's idle start time. It can be seen that at the time 0, every machine (instead of work centre) is ready to take an operation. So, the first machine M1a is selected to set up the first scheduling state. Then, this method finds out the operations that are waiting to be scheduled on this machine at the current time. Two operations are found, operation Op1a1 and Op2a1, which are from two different orders and require the same work centre (work centre 1). Supposing due to priority reasons, operation Op1a1 is selected to be scheduled now on M1a. After finishing this scheduling, another scheduling state will be set up. In the same way, all the machines are checked according to the present situation and machine M1b is chosen on which to schedule next. At this scheduling

state, only one schedulable operation is found and it is scheduled on machine M1b, starting also from time 0. This process goes on until no other machine has schedulable operations at time 0. Then, it moves chronologically forward. Only after operation Op1a1, which finishes at time 4, is a schedulable operation Op1b3 found at machine M3. Because there is only one operation, it is scheduled to occupy M3 from time 4 to time 8. The other machines are also checked against this time but no schedulable operation is found. A similar situation happens at the time 5.5 on M2. Operation Op2b2 is found to be a schedulable operation and is scheduled on the machine from time 5.5 to 10. Pursuing the parallel method in this fashion, a complete schedule will be constructed for those two orders, as shown in Figure 3.3.

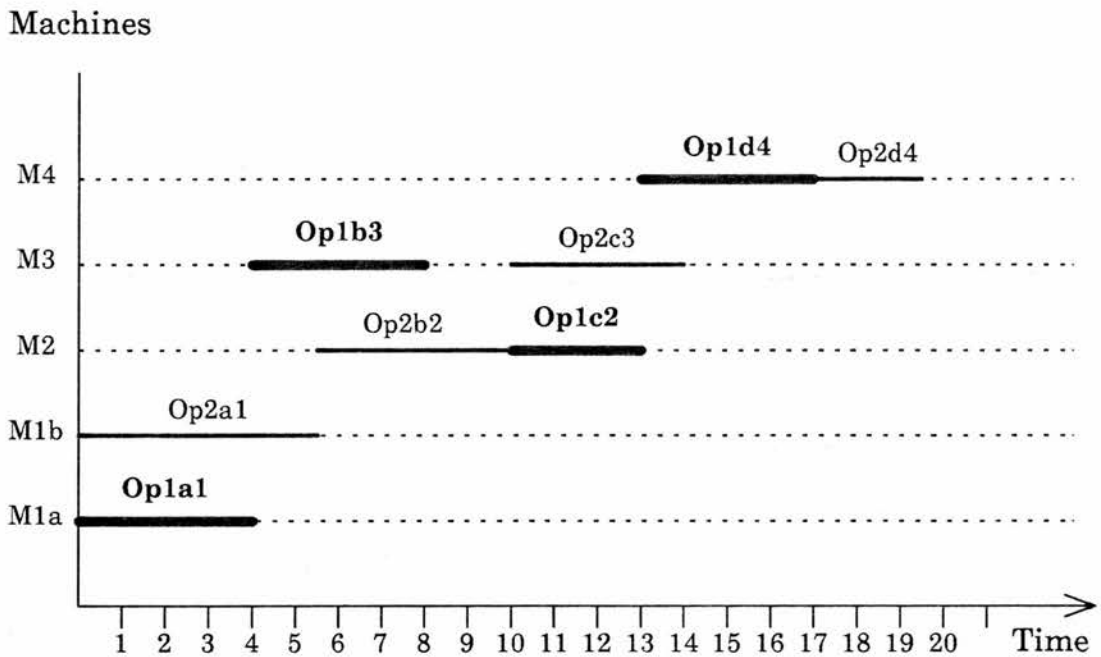


Figure 3.3 A schedule constructed with the use of the parallel scheduling method

If we compare this schedule with the one produced with the single order scheduling method in Figure 3.1, we can see that the overall schedule time has been reduced with the parallel scheduling method. The reason is that single ordering method restricts the consideration of the operations of other orders when scheduling the operations of the

current order such that more idle time is created. For instance, in Figure 3.1, part of the idle time before operation Op1c2 can actually be occupied by operation Op2b2 from the second order if parallel scheduling is used, as shown in Figure 3.3. But in single order scheduling, this operation (Op2b2) is not even considered when scheduling operation Op1c2, and when the system schedules the second order the time gap between the end time of operation Op2a1 and start time operation Op1c2 is not long enough for operation Op2b2. So it has to be scheduled after operation Op1c2 and that idle time will be left. Typically, this schedule will have some effect on the schedules of the subsequent operations, and in the end the last operation of order 2 finishes later (see Figure 3.1).

3.2 Sequence, Time and Machines

In scheduling based on existing resources, there are basically three important factors that decide a final schedule. The first one is the sequence of the operations on each machine. The second one is the time at which each operation starts or finishes. The third one is on which machine an operation is scheduled (in each work centre there is normally more than one machine).

These three factors are not independent of each other. Instead, they are inter-related by constraints. Although the sequence of operations seems to have nothing to do with the time when each operation starts and neither of them has anything to do with the machine on which an operation is scheduled, the constraints link them together. For example, in Figure 3.4, three orders need to be scheduled (their due times are also displayed). It is to satisfy the due date constraint and reduce idle time of resources that the operations are scheduled in this sequence on each machine, at these particular times and on their particular machines. Let us show how each of the factors influence a schedule.

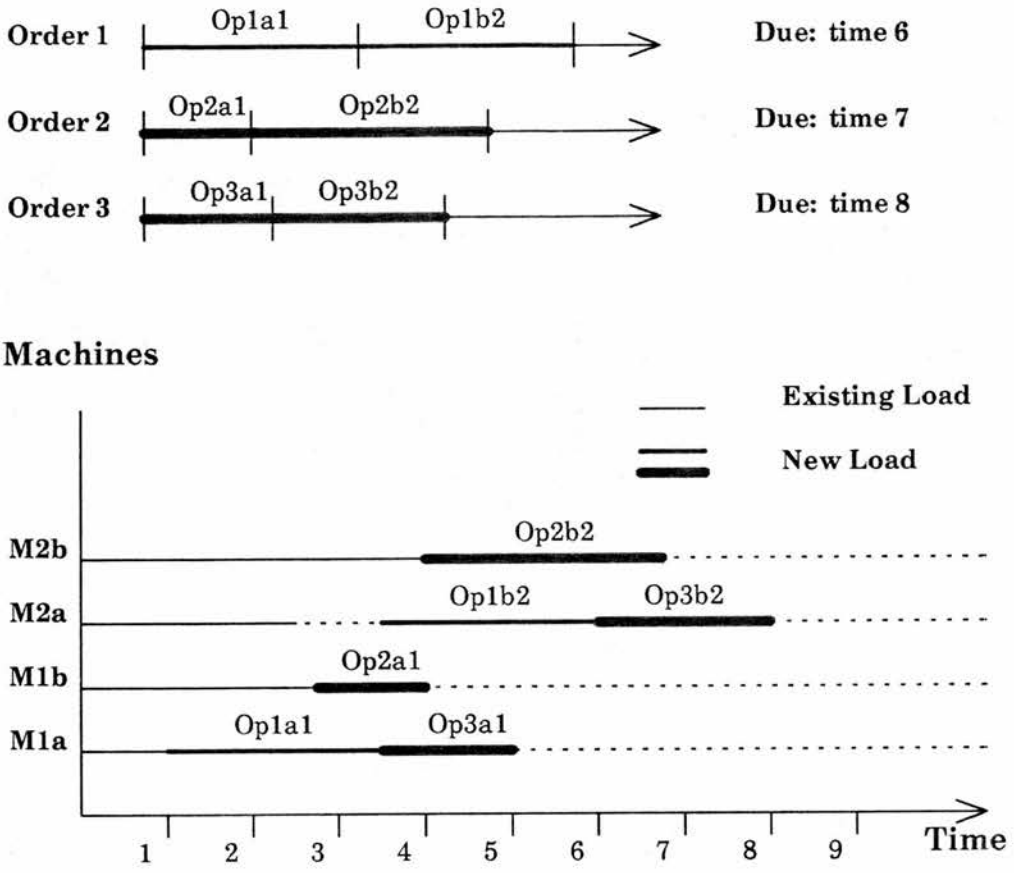


Figure 3.4 Sequence, time and machines

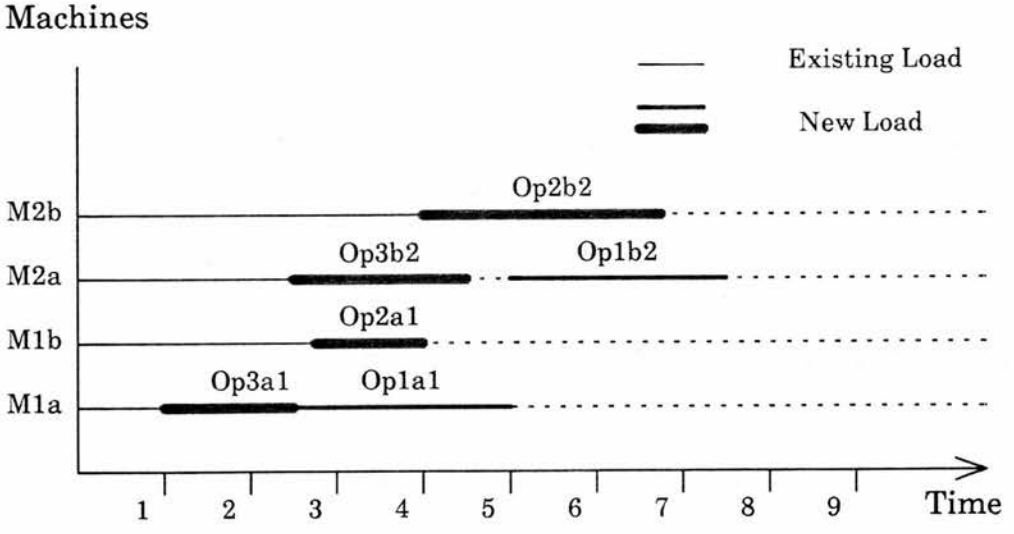


Figure 3.5 A changed sequence

Sequence is the most important factor that affects the scheduling result. This explains why scheduling is often called sequencing. For instance, if the sequence of the two operations on machine M1a in Figure 3.4 is changed to the one in Figure 3.5, the schedule due date constraint of order 1 will not now be satisfied.

Similarly, if the operation Op3b2 in Figure 3.4 is put after the operation Op2b2, the due date constraint of Order 3 will not be satisfied. So, the machine on which to schedule an operation is also important for constraint satisfaction.

However, time influences a schedule in a different way, and also plays a very important role in scheduling. A schedule is actually a time assignment to each manufacturing operation to specify when it should be performed and when it should be finished (i.e. each operation's schedule is a period of time occupying some specific resources).

The operations in a schedule can be delayed for any length of time if there is nothing to restrict it. In such a case, there will be unlimited choice of when an operation can be scheduled. However, there are many constraints that exist and almost all the constraints are expressed by time, such as due date and work-in-process. It is for their satisfaction that the operations are compressed together so that they are connected in time with each other as much as they can be. As a result, a problem with a single operation in the middle of a established schedule will typically influence the schedules of the other operations to cause their schedule times to be changed. In more serious situations, it may change the sequence of many operations in the schedule as the operations are also linked by precedence constraints.

Furthermore, scheduling is concerned with shared resources and the capacity of a shared resource is represented by how much time of the resource is available in a certain time period. Time progresses and cannot be reclaimed, therefore if a resource is not used during a period of time, this time period of capacity of this resource will never be reclaimed either. This is different from the consumable resource case.

Time can also solve some conflict problems. In scheduling an operation, if two constraints conflict with each other for a time period but both constraints have to be satisfied in order to proceed with scheduling, delay of the operation may solve this problem. For instance, if an operation is ready to be scheduled on a machine, but the tool this operation requires is not available at the moment (as it may still be used by another operation), then this operation may wait until both the tool and machine are available.

3.3 The Underlying Scheduling Problems

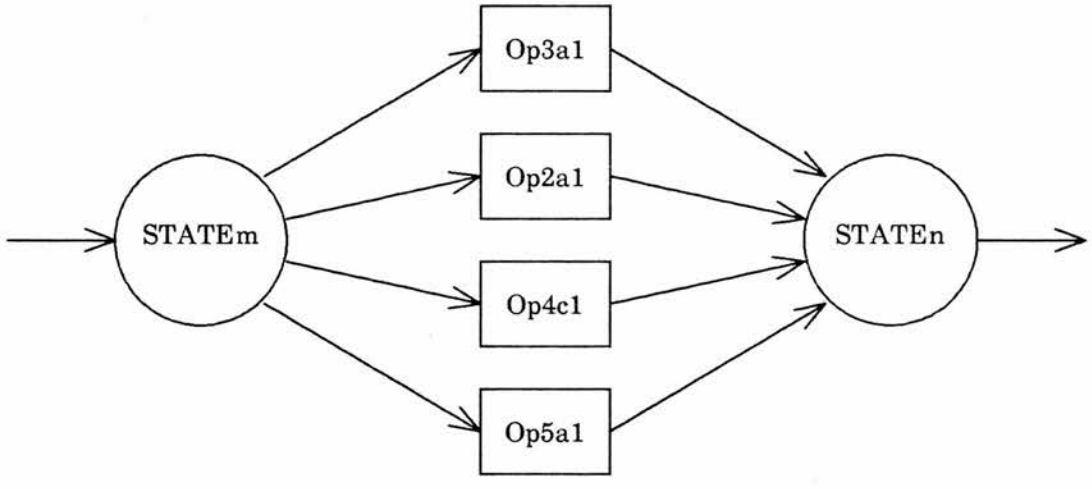
3.3.1 Scheduling Resource Problems

As discussed in the last chapter, scheduling is a NP-hard problem. Many approaches developed to date only have limited success in dealing with it. Before proposing a new approach to perform the scheduling task, in this section we take a close look at the scheduling process in the parallel scheduling method.

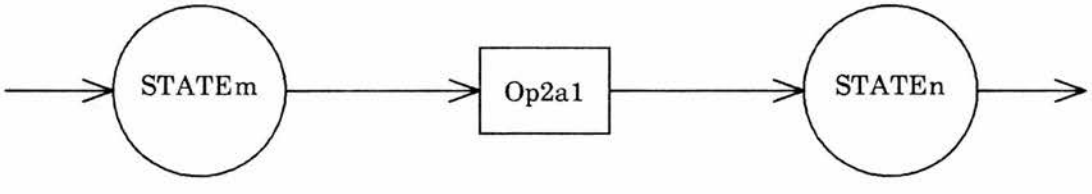
At a scheduling state, consisting of a machine and a time instance when the machine becomes idle and ready to take a new operation, an operation will be chosen to be scheduled on this selected machine starting from this time instance. In choosing such an operation, two levels of activities can be identified. The first level is to find out all the operations that are queuing at this machine, i.e. to check the ordering relationship (precedence constraint) between operations in their orders (a process of finding a set of operations which have their predecessors completed). This is the choice generation level.

Typically after this activity there is substantial freedom left for the final decision making as there is usually more than one operation that will be found at the first level (a set of operations) competing for the same machine. In other words, there is more than one operation that can be performed now in terms of time on this machine. In nonlinear planning [40], these operations will be in parallel (see Figure 3.6 (A)) between two states, STATEm and STATEn. But here the resource is limited; a decision has to be

made as to which operation should be chosen to be scheduled. This is shown in Figure 3.6 (B) (The lines with arrows in Figure 3.6 do not signify that the states are on the same machine but only indicate a flow from one state to another. State STATEm and state STATEn could be on different machines). This justifies the need for a level of choice making, the second level of choosing an operation to be performed from the set of operations generated by the first level.



(A) Scheduling with no resource limitation



(B) Scheduling with resource limitation

Figure 3.6 Scheduling decision making

As the objective of scheduling is to satisfy various constraints according to available resources, obviously, the criterion for making such a decision will be the constraints' satisfaction. Thus, the most important thing to do at this level is to assess the constraints' satisfaction on each alternative choice or candidate operation, and select the most favourable operation from the set of operations to be scheduled according to the constraint evaluation results.

However, in scheduling, many of the constraints, such as due date and work-in-process, express preferred final results. Their satisfaction can only be measured after scheduling has finished. So, in order to make a good choice, this level needs to effectively predict these constraints' satisfaction results. As there are many orders to be scheduled, which in turn have many operations, scheduling at the earlier stage or scheduling of the earlier operations will not know what will happen to the schedules of the rest of the operations. Such prediction is very difficult.

To be more specific, these orders or operations typically compete with each other for the same resources. This causes problems and many of these problems do not appear when the current scheduling is being performed at a local state. Instead, they normally manifest themselves after some or all of the scheduling has been done. Then, it may be too late to resolve them.

So, the heart of the problem is that the resources restrict the satisfaction of constraints. Figure 3.8, 3.9 and 3.10 show how resource capacity influences a schedule. Let us suppose that there are two orders that need to be scheduled (Figure 3.7). In Figure 3.8, there are sufficient resources available so that no operation will need to wait. In Figure 3.9, Work centre 2 only has one machine M2 and operation Op2b2 has to wait until operation Op1b2 has finished. In Figure 3.10, each work centre only has one machine and operation Op2c3 also has to wait. These three situations produce different scheduling results. Figure 3.8 is the best and the Figure 3.10 is the worst.

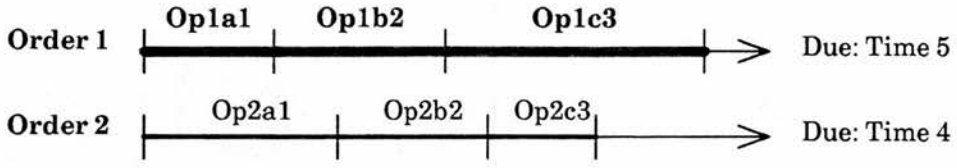


Figure 3.7 Two orders to be scheduled

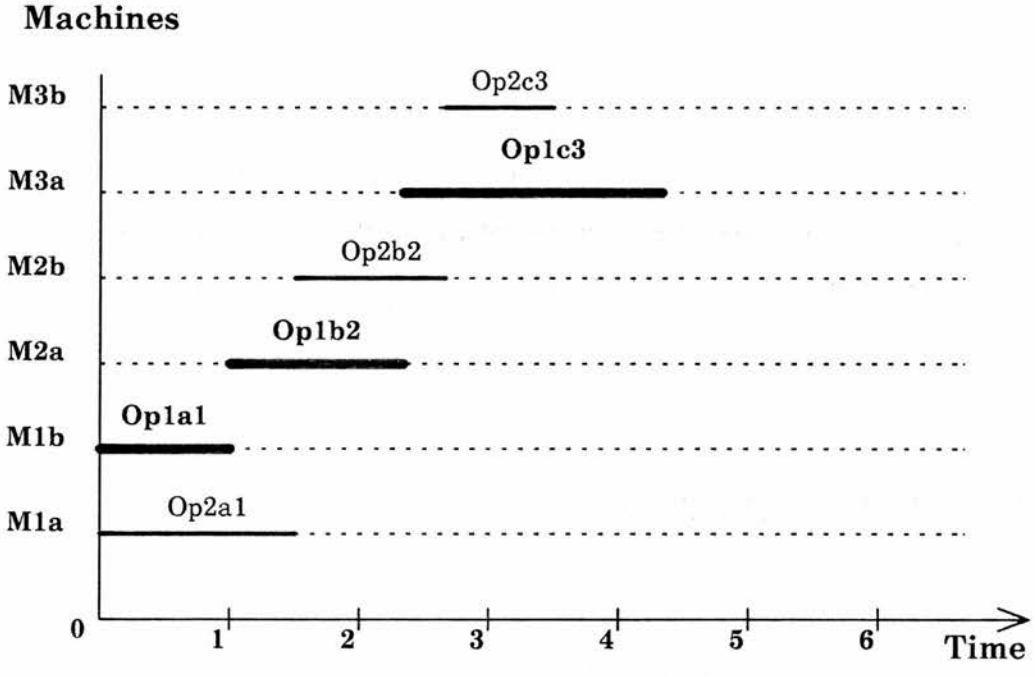


Figure 3.8 A schedule without resource problems

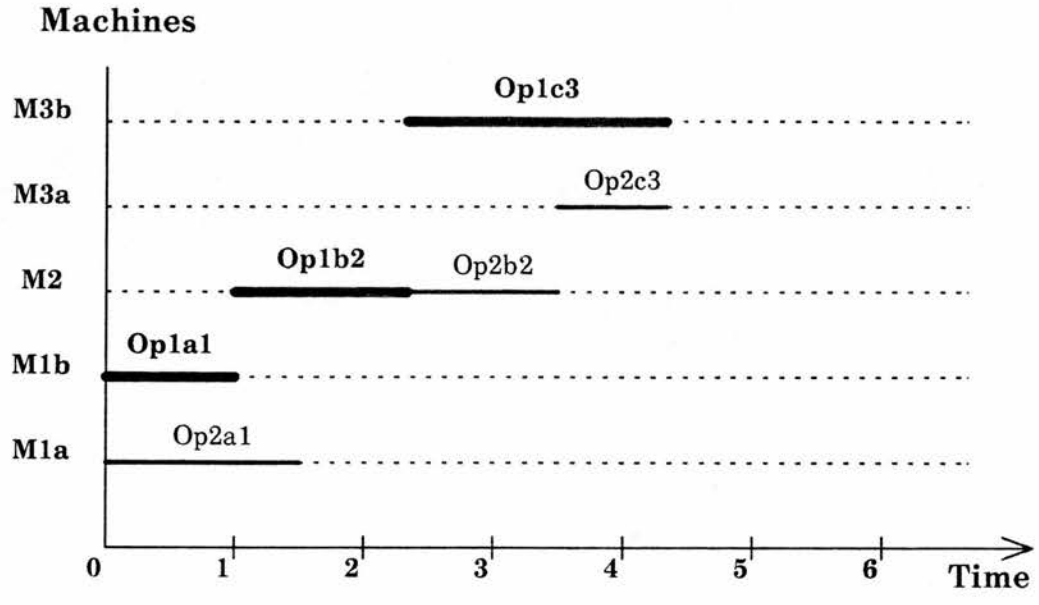


Figure 3.9 A schedule with less resource

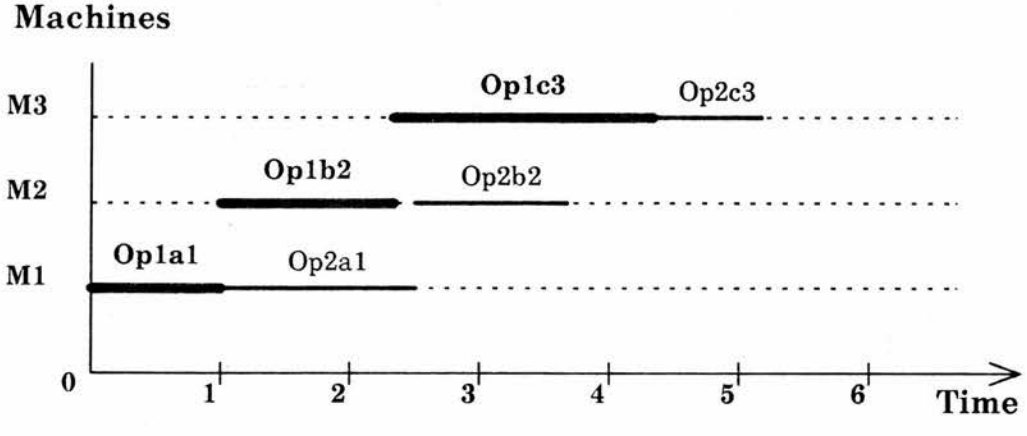


Figure 3.10 A schedule with even less resource

Two conclusions to be drawn from the proceeding discussion:

(1) As the capacities of the resources reduce, the problems become more severe

In Figure 3.8, the operations are performed in parallel and no waiting time is involved with any of the operations in both orders. But in the second situation, shown in Figure 3.9, there is only one machine in work centre 2 and consequently, operation Op2b2 has to wait until operation Op1b2 has finished. In Figure 3.10, the situation is even worse. None of the operation can go in parallel with any other operation. More time is therefore taken to finish these two orders.

As there is a due date associated with each order, the results can be evaluated according to the satisfaction of this constraint. There is no problem with the first situation. But in the second situation, the due date of order 2 will not be met. Nor will it be in the third situation (order 2 is finished much later than in the second situation). In real situations, there are many other constraints which need to be considered. The problem will become even more acute and a whole range of problems could be created.

(2) It is the limitation of resource that introduces the need to make decisions.

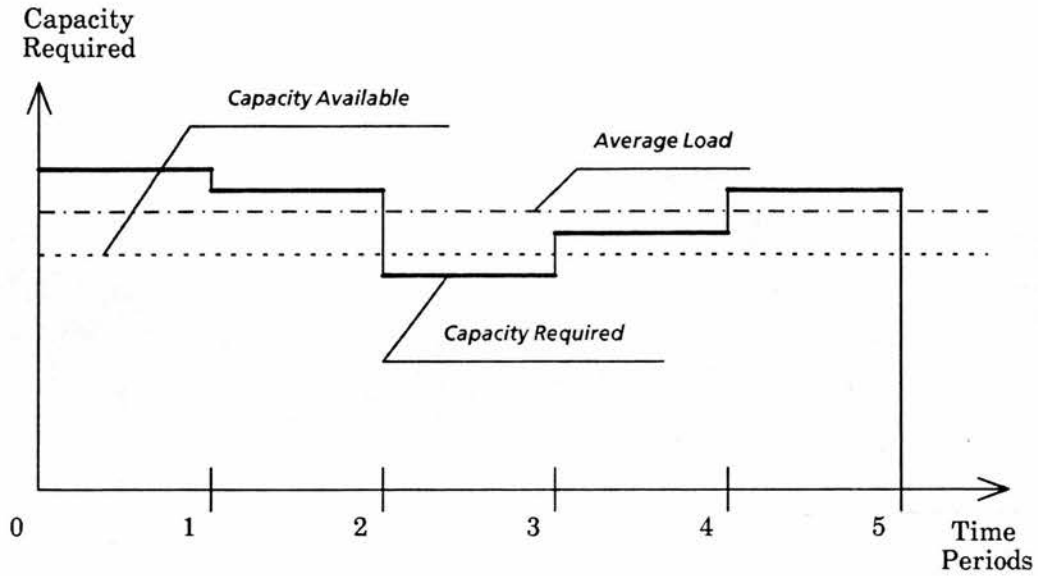
In the situation of Figure 3.8, there is no need to make any decision about the sequence of the operations (or which one first and which one next) as there are sufficient resource available to take every operation right after its previous operation has finished. In such a case, the orders can be finished in the least amount of time. However, in Figure 3.10, there is only one machine at each work centre. At machine M1 and time 0, there are two operations waiting (operation Op1a1 and operation Op2a1). Since a machine can only perform one operation a time, a decision has to be made to decide which operation to select to be scheduled first. The sequence of these operations will make a lot of difference to the final scheduling result and consequently, such decision making becomes

very important. The criterion for making such a decision here, as we mentioned earlier, is the due date constraint's satisfaction.

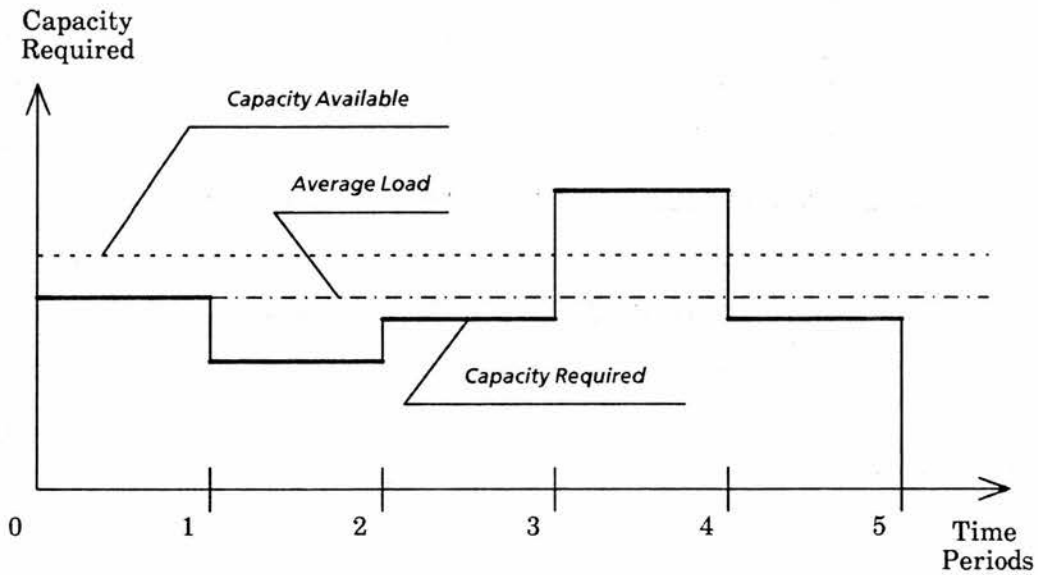
In job shop scheduling, the number of orders involved will not just be two or three as in the above examples, but hundreds. There are typically conflict situations which exist between constraint satisfaction and resource availability. These conflict situations are manifested as bottleneck work centres, which means that these work centres are overloaded during the time period concerned since there is too much demand for them. Consider work centre 2 in Figure 3.9. However, not all the machines will cause problems. Work centres 1 and 3 in Figure 3.9 do not cause any problems. Normally, in any situation, there are only a few bottleneck work centres in a scheduling situation that hamper constraint satisfaction.

Bottleneck work centres can be divided into two groups: global bottleneck work centres and local bottleneck work centres. A global bottleneck work centre will be overloaded during a long time period. A local bottleneck work centre is only overloaded during certain time periods, while in the other time periods it is underloaded. These bottleneck work centres will cause some operations which will be performed on these work centres to wait for a long time before they can be loaded, which in turn hamper constraint satisfaction. However the scheduling resource problems are not just these bottlenecks, otherwise there would be no way to tackle them. The underloaded time periods are also problems. So, balancing of the load can remove the local bottleneck work centres and reduce the effects of the global bottleneck work centres. Figure 3.11 illustrates the situations.

Bottleneck work centres influence the scheduling most, but balancing the load on problem work centres is not sufficient to deal with the problems. Sequencing of the operations on the machines of these work centres is also very important. As was mentioned before, the operation schedules are inter-related with each other in many ways, so, in order to have a good sequence on the machines in the bottleneck work centres, not only their schedules need to be carefully considered, but also the schedules on the other work centres.



(A) This work centre is globally overloaded.
(Global bottleneck work centre)



(B) This work centre has an overloaded period.
(Local bottleneck)

Figure 3.11 Bottleneck work centres

3.3.2 A Summary of the Problems

In the above, we discussed the resource problems in scheduling and their effects on a schedule. In order to balance the load of bottleneck work centres and properly sequence the operations, the system needs to first predict the problems before they occur. Then, it can try to choose operations to allocate the resources so that these problems will be avoided and their effects will be reduced. In other words, in order to achieve good constraint satisfaction, the system needs to predict the future effects of the current decisions, to try to avoid creating any problems and to reduce the influence of inevitable problems. The issue is how to effectively foresee the problems or how to predict what effects that the decision currently being made will have on the following schedules, at a local decision making state. Hence, besides considering the present situation, the scheduling decision making will also have to take into account the overall situation. Then the problem becomes one of how to achieve a global perspective of a local decision making point.

In more general terms, with constraints to be satisfied, scheduling will not just be a problem of checking available resource. More importantly, it is a problem of how to properly allocate the resources among those operations competing for them in the period concerned in order to have all the operations well organized and the constraints best satisfied. It is an organizational problem that concerns the current situation (or state) in the scheduling process as well as future situations.

Some may argue that backtracking can be used in the scheduling process if a problem is found at a later scheduling stage. Theoretically, chronological backtracking [1] can find the best result. But when the problems and number of alternative choices grow large, it will be impossible for such backtracking to manage. Dependency-directed backtracking [41] [42] is the most advanced method for choice making, but as it works on logical dependencies between operations, it would not be very useful at the second choice making level. As discussed above, the choices left from the ordering (precedence constraint) scrutiny at the first level are beyond logical dependency (or not strongly logically connected). Even if some problems can be found with dependency-directed

backtracking it is still not of great use for making changes inside a schedule and it could even cause looping. The reason is that job shop scheduling is a combinatorial NP-hard problem which normally involves more than one resource and more than one constraint. The changes made to one resource or constraint will typically have some influences on the others and possibly make them invalid. For instance, the system may find out that the cause of an order being overdue is that one of its operations has been scheduled too late. It goes back to this operation in the schedule and pushes it backward in time. Amongst other effects, this action also causes the tool that this operation uses to be pushed backward, which in turn may violate the resource availability constraint of an operation in another order and force this operation to be pushed forward. This forward pushing violates the due date constraint of this latter order. To fix this, the system may reverse the process, which causes looping. Of course such looping is not a serious problem and can be very easily stopped, but the point is that this is a kind of blind search. It may properly repair one problem but worsen others. Moreover such a repair process is normally extremely complicated, and it could be very difficult for a system to track. Perhaps the best thing to do is to forget the rest of the schedules after the failure and build them again.

As the resource capacity cannot be changed, based on the resource available there are also typically conflict situations between the constraints themselves. This will be discussed in the next chapter which introduces the constraint decomposition approach.

3.4 The Reinforcement Scheduling Approach

The last section discussed the detailed scheduling problems. This section will propose a way to deal with these problems. But before going any further, let us first sum up the problems identified so far.

(1) Scheduling is computationally NP-hard.

(2) There are typically conflict situations between resource availability and constraint satisfaction

(3) How to predict these conflicts or how to predict the effects of a local decision on a global solution

These three problems are not separate problems. It is because scheduling is a computationally NP-hard problem that it is impractical to try every possible schedule in order to achieve the optimal result. Hence, only a feasible schedule is asked for, and in such a case, problems (2) and (3) come to light. And, in order to resolve problem (2), problem (3) needs to be dealt with.

So, how to predict the impact of the local decisions on the global solution (or how to have a global perspective of the scheduling process) becomes the central problem in scheduling. We suggest that a pre-scheduling stage (which is different from the levels in decision making and is a complete planning process) should build a global view of the forthcoming events. The idea is that the system chooses a critical resource and builds a rough utilisation schedule as a basis for generating decision-making guidelines for constructing a detailed schedule.

Basically, this overall approach then consists of two stages of scheduling processes, the reinforcement stage of scheduling and the detailed stage of scheduling. The reinforcement scheduling stage (or pre-scheduling stage) produces a rough plan called the "reinforcement plan". Then, the detailed stage schedules again from the beginning and uses the reinforcement plan to help with its decision making. Such decision making will be a process of analysis and communication between these two stages, in which the aim is to reduce the effects of the global problems as well as the local ones.

There are three things that need to be further stated here. The first is that as mentioned before, a constraint decomposition approach is also proposed in this research and will be discussed in the next chapter. Its main objective is to consider different constraints in different scheduling processes, so different processes show different conflict situations. This section only discusses the reinforcement approach in order to deal with the problems between finite resource capacity and constraint satisfaction specifically, i.e. to

tackle the resource congestion and other related problems in order to satisfy the due date constraint.

The second is that the reinforcement approach can be not only used to tackle the resource problem in order to meet the due date, but also can be used to satisfy other constraints, such as work-in-process, in a constraint decomposition fashion. In the next chapter, along with analysing the problems between the constraints and presenting the constraint decomposition approach to deal with them, the reinforcement approach is generalised. Constraint decomposition can be seen as a supplement to reinforcement scheduling in dealing with scheduling problems.

Finally, there can be many ways to build a reinforcement plan, analyse it and use it for scheduling decision making in detailed scheduling. The following only provides a general description. We will later use the job shop scheduling system RESS-II to demonstrate an implementation of this technique.

3.4.1 Building a Reinforcement Schedule

Although there may be numerous distinct resources of various kinds for any one domain, some may never be used within a plan, others may be used only a few times. Normally, there is only one very important resource that needs particular care [8]. So, it is reasonable to concentrate more on this particular resource. The reinforcement approach starts by choosing this important resource to generate a rough plan based on it alone.

In our scheduling case, this primary resource is obviously the machine since every manufacturing operation has to be performed on a specified machine. Although an operation may also require tools, they normally have sufficient capacities or quantity in supply (as they are much less expensive than the machines). The reinforcement plan will be built based on the machine resource alone (here the "machine" is an aggregate

term for all the machines as typically in a job shop there are many machines which are designated for different functions).

However, in certain periods of time the less important resources can be heavily used in conflict with the main resource (the situations where the main resource is available but the less important resource required is not there). So, the system also records the usage of these resources during the process. But such recording does not make any commitment affecting their capacities. The actual recording can be done in the following way. During the reinforcement scheduling process, a decision is made to schedule an operation on a machine in a certain time period, i.e. the machine is allocated to a particular operation. Having committed to this decision, the system keeps a record about the usage of the tools it requires. For example, a decision is made to put the load of an operation in week 4. If this operation requires any particular tools, the system will keep a record of this information (in another source). Both of these operations constitute the reinforcement plan.

As resource allocation problems in scheduling are always restricted by constraints and involve two major components, resources and constraints, hence, there can be two basic ways of building such a reinforcement plan: one concentrates on the resources and the other concentrates on the satisfaction of constraints.

The first way would generate a reinforcement schedule (or plan) based on the important resource chosen. In such an method, the finite capacity of the resource is taken into account, i.e. if the machine is loaded to its maximum capacity no more operations can be performed on it. The process for constructing this plan is almost the same as the normal scheduling process as discussed in the first section of this chapter and using the parallel scheduling method or single-order scheduling method with some heuristic rules. The actual construction can be done either by the main system or a separate program which has the domain knowledge. However, this differs from normal scheduling in that availability of the secondary resources is not checked in this reinforcement schedule building process (merely their use recorded). There will be no attempt to positively try to satisfy the constraints or to evaluate the satisfaction of constraints on each alternative

choice. It is the heuristic rules employed that automatically satisfy these constraints or leave them unsatisfied. But, the rule chosen should have an aim, i.e. the heuristic rule used should be good for the satisfaction of a certain constraint. Since this stage is to provide the reinforcement schedule for satisfaction of due date constraints, so the rule used will also be good for the due date constraint's satisfaction. Otherwise, the reinforcement schedule generated cannot provide a good overall picture for the next stage of detailed scheduling. Consequently, the next stage of scheduling will have to make such considerable changes to the reinforcement schedule that it will not have any value for analysis.

If in this method both the primary and secondary resources are considered in the same way, it will be very difficult to decide what the problems really are. In most situations, the less important resources do not cause big problems. Considering them only wastes computation time and effort. Even if these resources have overloading problems, they will only be local congestions and can be balanced through local manipulation. The recording of the usage of these resources will provide sufficient information for making decisions for such local changes. However, the main resource could cause real difficulty in scheduling. Consider global bottleneck work centres (work centres overloaded over a long period) which put the due dates of some of the orders at risk, since some operations are delayed, waiting for others to complete, when their scheduled time is reached. Mixing these two kinds of resources produces confusion in detecting the problems. For example, the system may find a problem without knowing who the contributor is. In the worst case, the whole schedule may need to be changed. In detailed scheduling, tool congestion may be easily resolved, but it may leave the problems associated with the main resource very unclear and the resource problems subsequently found by the system unrealistic. For example, in reinforcement building, because of tool non-availability over a certain period of time, some urgent operations which need this tool and which otherwise should be able to be processed according to the main resource (the machine) will be delayed until the tool becomes available. Some other operations that are not urgent and do not use this tool will be scheduled in this period in order to take this period of resource capacity even though they do not necessarily need to be

scheduled. In the detailed scheduling, if this tool congestion is resolved, it will leave the schedule in the reinforcement schedule unrealistic.

The second method of construction ignores the finite capacity of this important resource chosen but concentrates on the satisfaction of a important constraint to generate a reinforcement plan. The other constraints will be ignored, except the constraints that have to be satisfied (such as start time of an order and precedence constraint of the operations, etc). The scheduling process loads the operations on their work centres in the time period concerned disregarding the work centres' overall capacities. In this method, the less important resources are considered in the same way as the important resource because the important resource scheduling is also a recording without actual decision making being involved (the finite capacity is not considered). The loading information of these recordings are kept in different places. The reinforcement plan constructed will give the system some idea of the loading condition of each work centre. (In RESS-II, this second method is used with the due date constraint as the most important constraint. We will go into detail in chapter 6 to discuss how this plan can be built.)

3.4.2 Reinforcement Schedule Analysis

Having finished the construction of the reinforcement plan, a post construction analysis is performed.

From the above description, we can see that some constraints and resource capacities are left unattended on purpose, so that the system will be able to find out the resource problems. This analysis determines whether any global resource problem exists, namely, the global bottleneck centres. As with the building of the reinforcement plan, there are also many ways to analyse the reinforcement plan. For example, as the two ways of building reinforcement will build different kinds of plan, so the analysis of them will also be different. To find the global bottleneck work centre, in the first situation, the system needs to access the waiting times of the operations on each machine to make this decision. However, in the second situation, summing up the loading in each period of

time on each work centre and comparing them with the capacities available will reveal which work centres are the bottlenecks. The result is passed to the next stage of detailed planning process.

Although the local bottleneck periods (temporally overloaded work centre) can also be detected from the reinforcement plan, they are not reliable as scheduling is highly context dependent. The manifested local high-contention areas in the reinforcement schedule may not actually appear in the detailed schedule or may not be in the same time periods. But this reinforcement plan does provide the anticipated load on each work centre and detailed scheduling will analyse this plan and its scheduling situation to enable it to decide when the local bottleneck work centres exist.

The reinforcement plan analysis can also be used for other purposes as well. For instance, it can provide management with the loading condition information so that they can make decisions to change the shop situation.

3.4.3 Detailed Scheduling

At this detailed stage of scheduling, the due date constraint is positively considered. The process concentrates on both predicting (or detecting) the resource problems and dealing with them, as unsatisfactory constraint satisfaction is mainly caused by the improper allocation of resources. If the resource problems are adequately handled, the number of constraints being satisfied or the degree of satisfaction of constraints will be automatically increased.

The less important resources and constraints are also taken into account at this stage. Instead of modifying the reinforcement plan to produce a detailed schedule, the system schedules again. This is a separate scheduling process which has nothing to do with the reinforcement plan except that the reinforcement plan provides the guideline for decision making. Here the system will try to reduce the effects of those global resource problems by taking extra care, taking earlier actions or trying to use alternatives, while

also dealing with the local problems encountered. The local bottlenecks are found by looking at the loading condition of the work centres in the reinforcement plan, which represented the anticipated load, and comparing it with the current scheduling situation. The whole scheduling process is a process of repeatedly analysing the current situation by searching through the partial detailed plan and consulting the reinforcement level of rough (but complete) plan in order to make decisions. The search through the partial detailed plan gives a current situation report on various resources. The search through the reinforcement plan finds out if any possible problems exist both now and in the future and sees whether they can be solved by proper allocation of resources to the actions. Associated with these searches is an evaluation mechanism to make the final decisions according to different constraints and resource problems.

After a particular decision has been made and resources have been allocated to the selected operations, the system updates the reinforcement plan to make it more reliable for future analysis. As the reinforcement building stage generates a rough plan, it cannot fully represent the detailed partial schedule. It is the purpose of this scheduling process to change the reinforcement plan in order to solve some potential problems. So, the decisions made in this process may change the reinforcement plan. As planning proceeds some expected problems may have been solved, some may not be as serious as were first thought and some new problems may emerge. Such updating will ensure that the reinforcement plan also reflects the current detailed planning situation.

We will go into more detail about how these are done in describing the RESS-II system in chapter 6, but for the time being, there are still a few points about this approach that need to be further stressed. The following gives a brief discussion.

(1) Scheduling on the machines in bottleneck work centres (both global and local) plays an important role in dealing with resource problems. Through careful sequencing of the operations and by finding alternative machines, these problems will be greatly eased. But this is just one aspect. The other, and even more important, aspect is how to make good decisions in the earlier stages of scheduling to ensure that these problems will not be created or their effects will

be maximally reduced. The most important thing about the reinforcement schedule is that it provides the detailed scheduling process with an expectation of what may happen in the future. Through analysis, the system will know what problems may appear later and if anything can be done about them. In other words, the system will predict what effects the current decision being made will have on the following schedules. More specifically, decision making in reinforcement scheduling involves look ahead procedures. At a scheduling decision making state, these procedures will enable the system to not only consider the current situation and the current loading condition in this time period, but also to look ahead to see whether there will be any problem in the near future and if they can be solved or eased by scheduling decisions now.

(2) Although global bottleneck work centres greatly influence a schedule, finding these resource problems and tackling them is only one part of problem detection and problem solving in reinforcement scheduling (they are not very difficult to decide). The local bottleneck periods of work centres also contribute a great deal. These resource problems are more subtle and difficult to decide on. Their detection is done by analysing the updated reinforcement plan.

(3) There are different ways to deal with both global bottleneck work centres and local work centres in different situations. For instance, in RESS-II, these problems are represented as rules. Each rule describes a problem situation and the way in which the problem can be solved in such a situation.

(4) In the above, we stressed the resource problems and their influences on scheduling. However, dealing with these resource problems does not take into account the nature of each individual order and their relative importance among other orders. It does not consider the local preference constraints either, which express the preferred choice among alternatives at a local choice point. Obviously, these constraints will also affect scheduling decisions, and hence should also be take into account. This completes the factors that need to be considered in the scheduling decision making.

(5) When considering the nature of an order among other orders at a local scheduling state, it is very difficult to use the constraints themselves to make judgements. For instance, since there are many operations competing for the same resources, there is no way to predict precisely when an order will be finished at a earlier scheduling state. So, they should be reflected by some other values in the scheduling process. For example, in RESS-II, the due date constraint of an order is represented by urgency and importance of the order, which express a priority value of an operation (from this order) among the other competing operations (from other orders).

(6) The reinforcement approach is not an approach that prunes the search space to a size which the system can explore without difficulty. Neither does it rely on heuristic rules, such as dispatching rules, to arrive at a schedule. Instead, it tackles the problem differently; it tries to predict the possible problems and then tries to solve them.

(7) This reinforcement approach does however propose ways of dealing with the problems discussed at the beginning of this section. To summarise, the computational effort required is small as it is only the analysis of current situation and reinforcement plan which helps to make the scheduling decision. There is no need to try a large number of alternatives blindly in order to find a feasible result. The scheduling problems can be found and dealt with by inter-stage analysis and communication, which in turn will produce a good schedule.

3.4.4 Schedule Construction and Schedule Refinement

We mentioned that the scheduling process so far is mainly for the due date constraint's satisfaction. In fact although the detailed process, discussed above, focuses on the due date it also shows up the conflict situations between the due date constraint and the work-in-process constraint. It is still not the main scheduling process, but a further

problem identification stage. The next scheduling process, which is the real schedule construction process (the main scheduling), will take these two schedules as its reinforcements to construct a preliminary schedule, and afterwards, a refinement process is performed in order to satisfy more local preference constraints, which will then produce the final schedule.

This is the basis of a constraint decomposition approach, and will be discussed in the next chapter. We will also postpone discussion of schedule construction and schedule refinement until then.

3.4.5 Monitoring and Rescheduling

So far a schedule has been constructed by passing through all the scheduling stages mentioned above. This schedule will then be used to guide and control the production process in the real world job shop environment. Shop activities will follow this schedule, but in most of the situations it is very unlikely that the schedule will be executed step by step to the end without any interruptions until the tasks they control are finished. Normally, there will be some unexpected problems, such as machine breakdown and later expected finish time of operations, which happen during the manufacturing process and invalidate the existing schedule. If the schedule is to guide the production process, it needs to be updated. This is the execution monitoring and rescheduling function.

The problems in replanning will include the input of interruption situations and updating the existing schedule. The input can be done using a representation scheme for the problems that can be understood by the system. The rescheduling process will be almost the same as the schedule construction but only the unexpected situations are considered. Resource allocation and constraint satisfaction problems will still be there. A special requirement is that the replanning process needs to retain the existing schedule as much as possible. So the ability to analyse the current situation and consult the past plan to combine them together to make decisions is also exactly what is needed

in execution monitoring and rescheduling. Thus, the reinforcement scheduling approach also provides a central technique for execution monitoring and rescheduling. We will go into detail in chapter 7 to discuss these problems and their impact on scheduling.

3.5 Summary

This chapter first discussed the scheduling problems in detail. In particular the conflict situations between the resources and constraints are identified. In order to deal with the problems, the reinforcement scheduling technique is proposed. However, this is still not sufficient for scheduling because there are also conflict problems between the constraints themselves, and they also have to be dealt with. This will be subject of the next chapter. However, the above discussion on the reinforcement approach is also based on the constraint decomposition approach, i.e. with only the due date constraint positively considered.

The initial version of RESS, namely RESS-I, only implemented this reinforcement scheduling but not the constraint decomposition approach. Consequently, the constraints problems are not well handled in its scheduling process.

Chapter 4

The Constraint Decomposition Approach

The last chapter discussed the problems between constraints and resources, and suggested that the general problem in scheduling (and other resource restricted domains) is that limited resources restrict the satisfaction of constraints and cause problems. So the emphasis was on dealing with the resource problems, i.e. balancing load and finding alternative work centres. However, as there is normally more than one constraint that needs to be satisfied in scheduling (this is perhaps also true in other domains), based on the existing resources, there are typically conflict situations that exist among the constraints themselves (i.e. these constraints will compete with each other for the same resources for their satisfaction). Such conflict situations also hamper the satisfaction of constraints. This chapter will focus on this issue and go into detail to discuss how each individual constraint can be satisfied along with the others.

As with the general conflict situations between constraint satisfaction and resource availability, it is equally difficult to predict and deal with these problems at a local state as most of them only appear when some or all the scheduling has been done. Therefore, a global perspective is also the main requirement. So, the reinforcement approach will

help. But it is not sufficient. After further discussing the constraints and their conflict situations in the scheduling process, it is suggested that with many constraints to be satisfied in scheduling, considering all of them in a single process will be very ineffective. Hence, the constraint decomposition approach is proposed. This method tries to consider different constraints at different scheduling stages (each stage is a complete scheduling process which produces a complete schedule) to show the conflict situations among them and to deal with them. Through using results from previous stages as the reinforcement schedules, the later processes, which try to consider the later constraints, will be able not only to predict the future problems at a local state but also maintain the previous constraint satisfaction results. These two approaches do not conflict each other. Instead constraint decomposition supports the reinforcement approach and the reinforcement approach makes constraint decomposition possible.

4.1 Constraint Analysis

In chapter two, we presented the scheduling environment, in which the schedule constraints were identified, classified and their influences on the scheduling process were explained. But, that classification of constraints and the explanations given were in a summary form, and is not sufficient for describing and dealing with the constraint satisfaction problem. In this section, we will go into detail to further discuss these matters, in which the shop constraints are first classified according to the nature of their satisfaction (rather than their functions) and then analysed according to their influences on the scheduling process and the schedule.

Since constraint satisfaction is the objective of the scheduling process, i.e. the central task of scheduling is to allocate limited shop resources to attend to a large and diverse set of scheduling constraints, this section to some extent forms the basis for the approaches of dealing with the scheduling problem. More specifically, it forms the basis for the constraint decomposition approach.

4.1.1 Job Shop Constraints and Their Classification

The following lists the shop constraints that were presented in chapter two. It is repeated here for reference.

- Organizational goals or objective constraints
  - (a) Meeting delivery due dates
  - (b) Minimizing work-in-process time
  - (c) Maximizing resource utilization
  - (d) Maintaining shop stability

- Local constraints
  - (e) Set-up
  - (f) Machine preference
  - (g) Job importance
  - (h) Urgency

- Technical constraints
  - (i) Machine
  - (j) Tool
  - (k) Precedence Constraint

This classification is according to their functions in the manufacturing production process, and does not give a clear picture about how they actually influence the scheduling process and the resulting schedule. Hence, the constraints in the same group can be quite different both in relation to the nature of their satisfaction and their impact on the scheduling process. Now we classify them according to these criteria, and in such a fashion, they can actually be divided into three groups: preference constraints, causal constraints and priority constraints.

(1) Preference constraints

Preference constraints are the ones whose satisfaction can be relaxed during the scheduling process, i.e. it is not strictly necessary to satisfy them. In problems

like scheduling there are so many constraints that need to be attended to in a resource limited environment. It is very difficult for a system to satisfy every constraint during the scheduling. Sometimes such an aim could even be impossible because there is not enough resource available or there is too much load requirement on the resources that it has exceeded the capacities of these resources. Hence, after a schedule has been constructed, typically, some of the constraints have not been satisfied (but, they cannot be those constraints that have to be satisfied in scheduling such as technical constraints). (a) to (f) above are the preference constraints.

Preference constraints can be further subdivided into global and local preference constraints. Global constraints express the desired final results. Their satisfaction can only be judged after the scheduling has finished, i.e. only the final results can show whether constraints have been satisfied and to what degree they have been satisfied. So, only certain operations need to be checked in order to evaluate satisfaction of the constraints. There is normally no restriction on the schedules of the rest of the operations (i.e. these will not take part in the evaluation of constraint satisfaction). For example, the due date constraint of an order requires the order being finished when due. It does not care where each operation is scheduled and when it is scheduled. Only the schedule of the last operation of the order tells whether this order is finished within the due date or not. Also, the work-in-process time constraint of an order cares only about the time when the first operation is released to the shop and the time when its last operation is finished. As for the operations in between, they will not be taken into account. However, this is not to say that the scheduling of these operations will not influence the satisfaction of the global preference constraints. In fact, as all the operations are linked by constraints, the schedules of the earlier operations will have some influence on the schedules of the later operations. From these points, there are two conclusions that can be drawn. The first is that the satisfaction of these constraints should be considered globally, i.e. the scheduling of the earlier operations should take into account the possible situations that could happen to the later operations to ensure that early

decisions will not create awkward situations for later operations. The second is that the operations can be manipulated in the schedule as long as they will not move the critical operations out of the range of these global constraints' satisfaction. In fact, even the critical operations can be shifted around as long as global constraints are still satisfied. The objective constraints above are the global preference constraints.

(The reason that the objective constraints are regarded as preference constraints is that they are not necessarily satisfied in the scheduling process. In most of the situations, some of these constraints will not be met but the schedule will still be used. For example, the due date constraint, though its satisfaction is very important, may not be satisfied for some orders.)

Local preference constraints express the preferred choices among alternatives at local scheduling decision making state. Their satisfaction will improve a schedule, but they are not as critical as the global ones. For example, if two similar operations which require the same or similar set-up tasks can be put next to each other on the same machine, the second operation does not have to do the set-up again. In such a case, a set-up time period will be saved which will in turn reduce the load time of an operation spent on a machine. (e) and (f) above are the local preference constraints.

(2) Causal constraints

Causal constraints define what conditions must be met before an operation can be performed. They are the constraints that have to be satisfied, i.e. their satisfaction is strictly required. Precedence constraints and resource availability constraints above (i.e. (i), (j) and (k) on page 87) are causal constraints. These constraints are set up by the process planning engineer who does the job of prescribing a process routing for each required part in order to transform raw material into a finished product. If these constraints or

requirements are not followed in the manufacturing process, the product cannot be produced or it cannot meet the required technical specifications.

The precedence constraints are actually the same as the ordering relationships in least commitment planning, expressing a precedence among the operations (or actions). The resource availability constraints are requirements for resources, i.e. they express that an operation has to be processed on a certain machine with the presence of certain tools.

(3) Priority constraints

Since there are normally many orders that need to be processed in scheduling, something is needed to express a priority value of one order among other orders during the scheduling decision making. This is the task of this class of constraints.

As mentioned in chapter 3, they, to some extent, can also be interpreted as representing the objective constraints at the local state, such as the due date constraint (because there is no way to predict precisely when an order can be finished).

After this grouping, the constraints should be presented as follows.

- Preference constraints
  - (1) Global preference constraints
    - (a) Meeting delivery due dates
    - (b) Minimizing work-in-process time
    - (c) Maximizing resource utilization
    - (d) Maintaining shop stability
  - (2) Local preference constraints
    - (a) Set-up
    - (b) Machine preference

- Causal constraints
    - (1) Ordering constraints (precedence constraints)
    - (2) Resource constraints
        - (a) Machine availability
        - (b) Tool availability
- Priority constraints
    - (1) Job importance
    - (2) Urgency

The constraints that are going to be discussed in the constraint decomposition approach are only the preference constraints, which are also the constraints that influence the performance of a factory in terms of production management or cost effectiveness. The ordering constraints and the resource availability constraints will not be analysed because they have to be satisfied anyway and have been discussed in chapter three. Nor will the priority constraints (which only express a priority value of an order at a scheduling state) and they will be taken into account during the decision making.

4.1.2 Constraint Satisfaction and Conflict Situations

With so many constraints involved, each of the constraints will not act independently of others during the scheduling process. Instead, a constraint will typically influence and interact with other constraints. In this research, we advocate predicting the future effects of the current decisions. So, understanding the constraint relationships and the constraint interaction situations becomes essential for a good scheduling result. As the constraints are of different kinds, the nature of such relationships and such conflict situations will vary a great deal. Some conflict situations need compromise decisions, while some others require all constraints involved to be satisfied (i.e. none can be relaxed). In this subsection, we identify and explain these situations and their impact on the scheduling process and the resulting schedule.

Scheduling involves an environment where many orders need to be scheduled and each order has its constraints to be satisfied. So, there are two perspectives with which to

consider a constraint associated with each order. One is to consider it within the order and the other is to consider it among other orders or with respect to the overall situation. For instance, to consider a due date constraint within an order is to ensure that the due date of the order will be met on completion of scheduling, while to consider it with the other orders (which also have their due dates associated) is to check its priority among other orders, as all these orders are competing with each other for the limited resources in order to have their due date constraints satisfied. In such a context, some orders are more important than others. So, the due dates of these more important orders should have higher priorities than those of less important orders to be satisfied. (To some extent, this global consideration of constraint satisfaction can also be interpreted as the same constraint of different orders interacting with each other.)

When we say that the constraints interact with one another, it does not imply only negative interactions. There are also beneficial (or positive interaction) situations and the situations where the constraints involved do not appear to have anything to do with each other. Negative interaction is the conflict situation, where the satisfaction of some constraints will negate or hamper the satisfaction of the others. In such situations normally a compromise decision will be needed. However such negative interaction does not happen all the time.

Positive interaction means that the satisfaction of one constraint will help the satisfaction of the others. For example, maximizing the resource usage will help the due date constraint satisfaction. Also, the constraints within an order may appear to help each other. Consider due date and work-in-process. If the due date is best satisfied in the scheduling then the work-in-process will not be too long.

In the rest of the situations, the constraints involved do not appear to have anything to do with each other, i.e. they can be satisfied in parallel. Let us take the due date constraint and work-in-process constraint as an example again. Although short work-in-process time will make due date very difficult to meet as the order will be very inflexible, within due date satisfaction range work-in-process time can be shortened or increased as much as it can be. The meeting of due date does not necessarily mean

work-in-process time is very short and very short work-in-process time does not mean the due date is met. In the first instance, an order might be started too early and in the second instance, an order might be started too late (or because of some other reasons as discussed below in analysing the constraint conflict situations).

However, it is still the conflict situations that affect schedules most and they deserve further discussion. The following lists the various conflict situation types and sets up the background for introducing the constraint decomposition approach.

- Constraint conflict situations among the causal constraints: Such conflicts would have to be resolved in any situation as the relaxation of any constraint involved means that the schedule constructed is unacceptable. In our approach, instead of resolving such conflicts, they will not be created in the scheduling process. For example, a situation could exist where the ordering constraint and the machine resource availability constraint are in conflict with each other, i.e. the resource is available but there is no operation which is waiting to be performed because none of the operations' ordering constraint can be satisfied or none of the operations' previous operations have been completed. This is handled in the choice generation level which finds the ready operations by checking whether any operations are waiting. Only those ready operations will be considered in the decision making in order to choose an operation to be scheduled. If there is no waiting operation, the machine will be idle. This prevents the conflict situations from happening.

- Constraint conflict situations between preference constraints and causal constraints: In this kind of situation, it is always the preference constraints which are relaxed since the causal constraints must be satisfied. For instance, when an order becomes very critical (due date is approaching), but there is no machine available to process it. In such a case, there is nothing that can be done except to wait until the machine is free.

- Constraint conflict situations among global preference constraints: This is a conflict situation where some global constraints are satisfied, the others have to be relaxed. For example, trying to satisfy the due date constraints and sacrificing the work-in-process constraints. As there are many operations (from different orders) that need to be scheduled at a state, normally there is a period of waiting time between two successive operations in the same order. Two reasons cause this wait. The first reason is that an operation is ready to be processed on a machine but the machine is still occupied by another operation. The second reason is that there may be more than one operation waiting to be performed at a machine at a particular point in time. One operation is selected and the others have to wait. Typically, at the bottleneck work centres, the waiting list is very long so that some of the operations will have to wait for a long time before they are performed. These long waiting times at the bottlenecks cause long work-in-process times. They are the conflict situations between the work-in-process constraint and the due date constraint (caused by the resource, of course) because the satisfaction of the due date of every order was the influence behind the decisions to schedule the operations on the bottlenecks. (Right shifting of the operations before the bottleneck machines will reduce the work-in-process time.) As mentioned above the constraints in the same order seem to help each other, so here the conflict situations are caused by the due date constraints of certain orders conflicting with the work-in-process time of other orders.

- Constraint conflict situations among the local preference constraints: In the scheduling process, when the system tries to choose an operation for the machine selected (which sets up a scheduling state), some operations may find that their set-ups are the same as the one that has just been scheduled on this machine (so that a set-up time period can be saved) but this is not their preferred machine. Similarly, some other operations may find their preferred machine is this machine but their set-ups are very different from the set-up of the operation that has just been scheduled on it. Of course, there are also some other

situations where both the machine preference and the set-up constraints can be satisfied at the same scheduling state.

- Constraint conflict situation between global preference constraints and local preference constraints: This kind of conflict is caused by the situations where the satisfaction of global constraints will negate the satisfaction of local constraints, or vice versa. For example, an operation has to be scheduled on a machine at a certain time in order to satisfy the due date constraint, but this machine may not be the preferred machine of the operation and its preferred machine's idle start time may be too late for the due date. Like the conflict situations among the global constraints themselves, this kind of conflict situations is also very uncertain since the satisfaction of global constraints is unclear. However, this means that such conflict situations are not necessarily real when they emerge and there may be a situation where both kinds of constraint can be satisfied.

- Conflict situations between the resource problems and the constraints: These conflicts arise in situations where the constraints block the solution of resource problems, such as bottleneck machines. For instance, when a resource problem has been predicted but no fix can be found. These constraints are most probably the causal constraints as their satisfaction is the premise for any further actions.

Although in the above so many conflict situations have been discussed separately, they are not normally independent but typically connected with each other in some way. In other words, there may be more than one kind of conflict situations involved in one problem. However, they are still all related to the limited resources.

4.2 The Constraint Decomposition Approach

4.2.1 The Motive

In the situations where more than one constraint is involved, a compromise decision is often needed to reflect the consideration of different aspects of the problem. As the constraints typically conflict with each other as discussed in the last section, in order to achieve an effective compromise among them, reliable information about the satisfaction of every constraint involved, their interaction and the impact of such satisfaction on the later planning or scheduling process is essential. However, current techniques which deal with constraints only evaluate their satisfaction according to local information and a partial schedule, or what has been done so far. They do not consider what may happen next and what may be the impact of current decisions. They typically consider all the constraints in a single planning or scheduling process [31]. At any local decision making state, the system accesses all the constraints and rates each alternative choice according to its constraint satisfaction results. Such a rating is supposed to reflect a compromise among these competing factors and the most favorable choice is chosen to continue the process.

However in most situations, especially those involving global preference constraints and resources such as scheduling, it is very difficult to predict every individual constraint's satisfaction result at a local state as its satisfaction can only be measured after the whole planning or scheduling process has finished. For example, when the system is scheduling the first operation of an order, it is almost impossible to predict when the order will finish and whether it can be finished when due because the final result also depends on the scheduling of the other operations in the same order and the operations from the other orders (which are competing for the resources in order to meet their due date constraints). Although it is the resources that restrict the constraints' satisfaction and cause problems, the constraints will typically conflict with each other. These conflict situations also hamper the satisfaction of one or another constraint's satisfaction. It is equally difficult to foresee when and where such conflict situations

may occur. As a result, the system cannot know at the local decision making state what impacts the current choice will have on the rest of the schedules or what effects the current decision being made will have on the final constraint satisfaction result. So, the constraint satisfaction rating or the compromise decision made is very unreliable and even misleading.

In the reinforcement approach, an overall picture is built and used to help deal with this problem. But that is not sufficient as a reinforcement plan typically can only give a rough overall picture about one kind of problem. If it is used for more constraints, their satisfaction and the interactions will still be unclear. For example, in reinforcement scheduling discussed in the last chapter, the first reinforcement plan shows the problem between resource required and resource available in order to satisfy the due date constraints of all the orders. If it is also used for the work-in-process constraints, the decision made could be unreliable. The reason is that although the system could use the reinforcement schedule to find out the global bottlenecks, local bottlenecks cannot be decided and those operations that will be delayed at the bottlenecks are not clear since not all the operations to be performed on the bottleneck machines will be delayed. So, it is very difficult for the second stage of scheduling to predict the problems between the due date constraint and the work-in-process constraint. However, after the due date constraint has been considered in a process, it will be much easier to see the problem situations between them and when these problem situations may occur.

4.2.2 The Approach

With the problems discussed above, we need to propose another approach in addition to reinforcement scheduling to deal with them. For this purpose, the constraint decomposition approach is introduced. This approach is characterized by showing different scheduling problems with different scheduling processes (or different stages). The idea is that before the main scheduling could be done there should be some reinforcement schedules to show the problems between different constraints and their relationship in scheduling, then the main scheduling uses this information to make the

scheduling decision. Consequently, the decision made will be able to reflect a dependable compromise. (Note reinforcement scheduling also plays an important role here.)

The underlying basis for this approach is that as indicated in the last section, in most constraint satisfaction situations more than one position in the plan can satisfy a constraint or typically there is a range in which a constraint can remain satisfied. This is particularly true of the global preference constraints as they only care about the first and the last operations' schedules. Within this satisfaction range the schedules of all the operations can be manipulated in order to try to satisfy other constraints. Or, in other words, there may be some points in this range that the constraints which cannot be satisfied originally may also be satisfied. If a plan or schedule could be manipulated to reach such a situation, the result produced will certainly be better. We call this the positive way of satisfying constraints.

So, the essence of this approach is to find this satisfaction range and try to manipulate within it. But, it is very difficult to know what the satisfaction range is for a constraint when the system is actually trying to satisfy this constraint since it does not know what will happen in the future. In the constraint decomposition approach, however, the system tries to satisfy a certain constraint at certain scheduling stage while only considering the others opportunistically. After a constraint has been considered, it will be very easy to see what its satisfaction range is and how other constraints can be better satisfied within this range (we call this the conflict situation). To some extent, such an approach can also be interpreted as considering different constraints in different scheduling processes (or stages).

A important policy is that in considering the new constraints, the system assumes that the previous processes have done their best to satisfy those old constraints and new processes should maintain the previous constraint satisfaction result. There should be no compromise decision between different constraints in different scheduling processes (or stages). Later ones always have to be relaxed if they conflict with the constraints in the previous scheduling processes and if such conflicts cannot be resolved. The reason

for this is that if a compromise decision is made while asking the previous constraints to relax, there will be no picture for the later constraints' satisfaction to refer to since the problems shown in reinforcement schedules will have been changed.

Now, let us go into detail to analyse why such an approach is possible in scheduling.

(1) The conflict situations between global preference constraints can be exposed by different processes.

Since global preference constraints always address the final result rather than local decisions, many things may be changed locally without altering this final result, i.e. the system can manipulate a schedule in order to satisfy new constraints without losing the satisfaction of the previous constraints. This is one of the main ideas on which the decomposition approach is based. In scheduling there are two spaces that offer opportunities for such manipulation. One is between the finish time of an order and its actual due date. It results from the situation where the order finishes ahead of time or finishes before due time so that their schedules can actually be pushed forward in time as long as the due dates are not violated. The second is the waiting time period between operations (this has been discussed in the last section), and within these waiting periods, the global constraints remain satisfied.

The work-in-process constraint and the due date constraint conflict situations discussed in the last section is a good example. In this example, we can see that it is the long waiting time at the bottleneck work centre that causes the work-in-process time to be correspondingly long. There are two ways to reduce the work-in-process time. One is to push backward the operations which have long waiting times so that the waiting times are shortened. In this case, some of schedules on the bottlenecks will be changed, which may violate the due date constraints of some orders. The other is to push forward (in time) the operations which are before the long waiting operations. The waiting times are also

reduced. This method maintains the due date constraint satisfaction while also having the same effect on the work-in-process time.

The fact that different constraints normally have different importance values supports the decomposition approach. For example, in scheduling, the due date constraint is more important than any other and is considered first.

(2) It is difficult to consider the local constraints together with the global constraints.

It is our aim in the reinforcement approach to try to analyse the scheduling constraint satisfaction problems in a global perspective, but this is only limited to the satisfaction of the global constraints, not the satisfaction of the local preference constraints. In other words, those reinforcement schedules are built only for showing the problems among the global constraints and the resource problems but not for providing an overall picture for the satisfaction of local preference constraints. So there is no ground to consider the local constraints globally. In fact, it will be very difficult to consider the local constraints globally at the same time as the global constraints, and this may constitute another NP-hard problem.

It is also quite difficult for the system to try to positively consider both the global preference constraints and local preference constraints at the same time or in the same scheduling process. The reason is that the satisfaction of a global constraint can change a schedule a great deal, and if the local preference constraints are to be considered at the same time, the only thing that can be done to them is to produce a compromise satisfaction rating to join the global constraint's satisfaction consideration. In this case, because the local preference constraints have no ground to refer to they are not positively considered and consequently those constraints that can be satisfied are missed. Furthermore, the main problem of satisfying the global preference constraints could overshadow the local preference constraints' satisfaction and the global preference constraints' satisfaction could be disturbed by the local preference

constraints. For example, in trying to positively shorten the work-in-process time, the operations before the operations that use the bottleneck machines can be shifted forward (right) in order to reduce the waiting time at these machines. But the effect of such forward pushing can be so dramatic that the system will not be able to positively consider the local preference constraints at the same time because the system could not know what impact such positive consideration will have on the scheduling result. If the local preference constraints are considered only opportunistically, two sorts of rating will be produced. However, since the global constraints are always more important than the local preference constraints, it is often the case that the local preference constraints are relaxed. The final result will not be so good.

Finally, we cannot work the other way round, to satisfy the local constraints globally and satisfy the global constraints opportunistically, since the global constraints are much more important and they are the objectives of the scheduling or planning process. The local preference constraints only help to improve the result.

(3) Local constraints should be considered in the same process.

Let us suppose that there are N local preference constraints C1, C2, ...Cn that need to be considered in scheduling. There could be two ways to satisfy them. The first is to consider all these constraints in one scheduling process, i.e. seeking a compromise among them. At a choice making state each alternative choice is evaluated according to the satisfaction of these constraints. The result of such an evaluation can be expressed as a rating to be included in the decision making.

The second is to decompose these local preference constraints and satisfy each of them in a different scheduling process. The principle could be the same as that of satisfying the global preference constraints. In the first process, the system tries to satisfy constraint C1 and it produces the first schedule. In the second process,

it tries to satisfy C2 while also maintaining C1, and so produces the second schedule. However such a process is impossible because when the system either tries to satisfy the second constraint while preserving the first one, or make a compromise between these two, at a scheduling state, some local changes must be made to the first schedule. Although they are done in the second partial schedule, they cause the first schedule to be updated and altered to some extent. Unlike the global constraints, the satisfaction of these constraints depends on the local situation. The difference between the second partial schedule and the first schedule at the earlier stage will make the whole range of C1 satisfaction invalid as far as the second scheduling process is concerned (the global constraint satisfaction may not be changed though). Then the system will have to try to satisfy both of them in the second process. This happens to all the rest of the constraints. Hence, there should be only one process for all the local preference constraints.

Since these constraints depend on local situations, any further change will influence the outcome of their satisfaction. So, they will only be considered after all the global constraints have been considered, and the process of handling them only makes improvements upon the schedule produced from the consideration of all the global constraints.

To deal with these constraints in a constraint decomposition approach will be quite different from a simple compromise decision, although they are only considered in one scheduling process. Firstly, constraint satisfaction in such an approach is positive (which means that the previous scheduling result can be changed in order to satisfy more local preference constraints) rather than negative (by simply selecting the best suitable choice at a scheduling state without making any effort to create a situation for even better constraint satisfaction). Secondly, it must not affect the satisfaction of those global constraints, i.e. the changes to a schedule can only be made within their satisfaction ranges. Hence, the satisfaction of local preference constraints are treated in two ways. Regarding their interactions with the global constraints,

there should be no compromise decision if such interactions cannot be resolved, but only local preference constraint relaxation. But, among the local preference constraints themselves, a compromise decision is required to find the best operation to be considered.

To save computational time, sometimes it is possible to consider the local preference constraints in the monitoring and replanning instead of in the plan construction. Then, the schedule from the schedule construction only provides a basic configuration of the forthcoming events. The reason for this is that in plan execution there are frequent unexpected interruptions that occur which force the existing schedule to be changed. In such situations, the global constraints may stay unchanged but the local preferences are very likely to be altered.

(4) Reinforcement approach provides a way to separate those different constraints

When the system tries to positively consider some new constraints, it has to keep the constraints satisfied in the previous processes unchanged or with little change. Otherwise such separation cannot be justified. The reinforcement approach provides this facility, i.e. in the process of changing a schedule to satisfy the new constraints, the system consults the reinforcements to access the impact of such changes on the existing constraint satisfaction, and if they violate the existing constraints then the change will not be permitted.

Hence, the reinforcement scheduling has two very important functions in the constraint decomposition approach. Firstly, reinforcement schedules show up the problems between the constraints. By analysis, the system can find out these problems and then try to deal with them by avoiding them or reducing their effect. Secondly, it provides a facility to maintain the previous constraint satisfaction results in the later processes.

To round off this section, let us stress some points about this approach:

- For global constraints there are different reinforcement scheduling stages to show the problems between them. Each reinforcement schedule produced exposes the conflict situations between a new constraint (which will be considered next) and the previous constraints.

- For the local preference constraints there is only one process for all of them and they are only considered after all the global constraints have been dealt with.

- There is no compromise decision concerning constraint satisfaction between stages. Later constraints should be relaxed if they conflict with the earlier ones.

- During the process of positively considering some constraints, other constraints are satisfied opportunistically.

4.2.3 Generalising Reinforcement Scheduling

The reinforcement scheduling discussed in the last chapter focused on resource problems. It basically functions with two stages (or two scheduling processes). The first stage chooses a critical resource and builds a rough schedule (reinforcement schedule) as an overall picture of forthcoming events. It shows the possible global problems between resource capacity and resource requirement in order to satisfy the due date constraint. The second (detailed) stage schedules by analysing the reinforcement schedule to try to deal with these global problems and also the local problems encountered.

It can be generalised as a technique that schedules or plans with the help of some external plans (reinforcement plans) which are able to provide an overall picture of the situation. More than one plan can be used as reinforcement, such as in the main scheduling stage of RESS-II, two reinforcement schedules are used. By analysing such plans, the system will be able to predict the potential problems or conflict situations

among constraints and resources that the current scheduling (or planning) process may have to face. It then tries to avoid or reduce the effects of these possible problems.

4.3 The Complete Scheduling Process

The reinforcement scheduling stresses that external schedules are used to reinforce the decision making in the later scheduling processes, while the constraint decomposition approach stresses that one reinforcement schedule is not enough for showing all the scheduling problems, and advocates that more stages should be used for such purpose. To put the reinforcement scheduling and the constraint decomposition approach into a system, there will be three parts in the schedule construction, a reinforcement building part which has two stages of scheduling, a main scheduling part which has only one scheduling process and a refinement part which also has only one stage of scheduling. The two reinforcement scheduling stages provide two reinforcement schedules. The main scheduling process uses these two reinforcement schedules for its decision making and generates a schedule. After the schedule has been generated, it is refined to satisfy more local preference constraints.

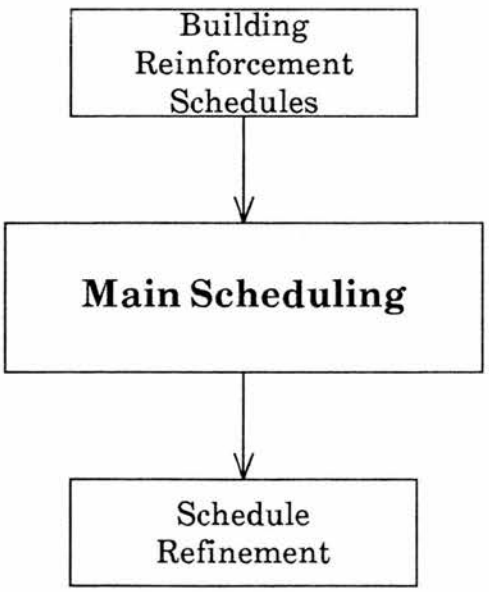


Figure 4.1 The complete schedule construction process

(1) Reinforcement building

There are two global constraints that need to be considered here: the due date constraint and the work-in-process constraint. The shop stability constraint will be considered during monitoring and rescheduling. Avoiding idle time will satisfy the resource utilisation constraint.

So there are two stages of reinforcement building. Actually, they are the two stages described in the last chapter. The first one showed the conflict problems between constraint satisfaction and resource availability, and the second one, along with trying to satisfy the due date constraints, showed the conflict problems between the due date constraint and the work-in-process constraint.

(2) Main scheduling

After the reinforcement schedules have been built, the system is ready to construct the real schedule by using these reinforcements in its decision making. The actual construction process is almost the same as the second stage of reinforcement building for showing the problems between the due date constraint and the work-in-process constraint, except that more information is available and more problems are considered. However, this stage still leaves the local preference constraints unconsidered.

(3) Schedule Refinement

This part considers local preference constraints, i.e. machine preference and set-up time. It is a process of local optimization in order to try to satisfy as many constraints as possible while avoiding interactions with the global constraints. By moving the operation around and using previous scheduling result as reinforcement, the system will be able to make better choices.

It corresponds to the discussion in (3) in the last section (trying to satisfy more local preference constraints). It is actually a refinement stage, in which the system refines the schedule that has been constructed in the main scheduling. The reason for calling it a refinement stage is that the system will only try to make some minor local changes to the earlier result in order to improve it and there will be no big alteration made. The difference between this refinement and the previous reinforcement scheduling is that the refinement actually uses the schedule constructed as its framework in the scheduling process, and only tries shifting the operations around locally without making significant change to the existing schedule. However, in reinforcement scheduling, the earlier processes only provide information for the main process about the constraint satisfaction and constraint conflicts. The main process will not use those schedules from reinforcement building stages as its schedule framework but just use them for generating decision making guidelines.

4.4 Summary

This chapter further discussed the constraints and their satisfaction in scheduling process. The main focus was on the constraint conflict situations. Detailed analysis suggests that with so many constraints to be satisfied in the scheduling, to consider all of them in a single scheduling process is very ineffective. Hence, the constraint decomposition approach is proposed, which is characterized by considering different constraint conflict situations in different scheduling processes. It is a technique following on from the reinforcement approach, without which the decomposition would not be possible as the inter-stage analysis and communication play an important role not only in providing the problems but also in helping to maintain the previous constraint satisfaction results.

With both the reinforcement scheduling and the constraint decomposition approach, the scheduling becomes a multi-stage process, within which each stage is a separate

scheduling. The earlier stages provide an overall picture for the forthcoming events and the later stages use this as the reinforcement or external help in their decision making.

This finishes the discussion of the approaches proposed in this research. The next three chapters will describe a scheduling system (RESS-II) which implements these techniques.

Chapter 5

Modeling the Scheduling Environment

A fundamental requirement for scheduling is a realistic model of the factory environment. This model will provide a hierarchical description of the production environment and its constraints. This chapter presents this model, and in addition defines the details of the representation adopted for the schedules to be generated and manipulated.

5.1 Factory Model

Job shop scheduling concerns many resources, constraints and activities. In order to perform this task, a rich underlying model of the scheduling environment is required. Such a model will not only represent the factory scheduling domain, but also provides a detailed system structure for scheduling itself.

Perhaps the most important aspect of this modeling is to formalise the internal relationship among different concerns. In other words, the model should capture the

internal organization of the factory shop floor from management decision making to production process so that detailed knowledge of all facets for scheduling, including orders, operation process routings, work centres, tools, constraints, etc. can be accessible in the right order at the right time in the scheduling process.

In RESS-II, a simple kind of frame-based representation is employed to describe these concerns and their relationship. The flow of the process is like this: orders from management with necessary data attached to them specify the required parts, and the required parts are manufactured in the factory shop floor. In order to produce a part, its process routing is required. This routing describes how this part can be manufactured through a set of operations, and what machines and tools these operations need. Figure 5.1 illustrates the organization model: orders for products from management get their routings to go to production, and the production is supported by work centres and tools.

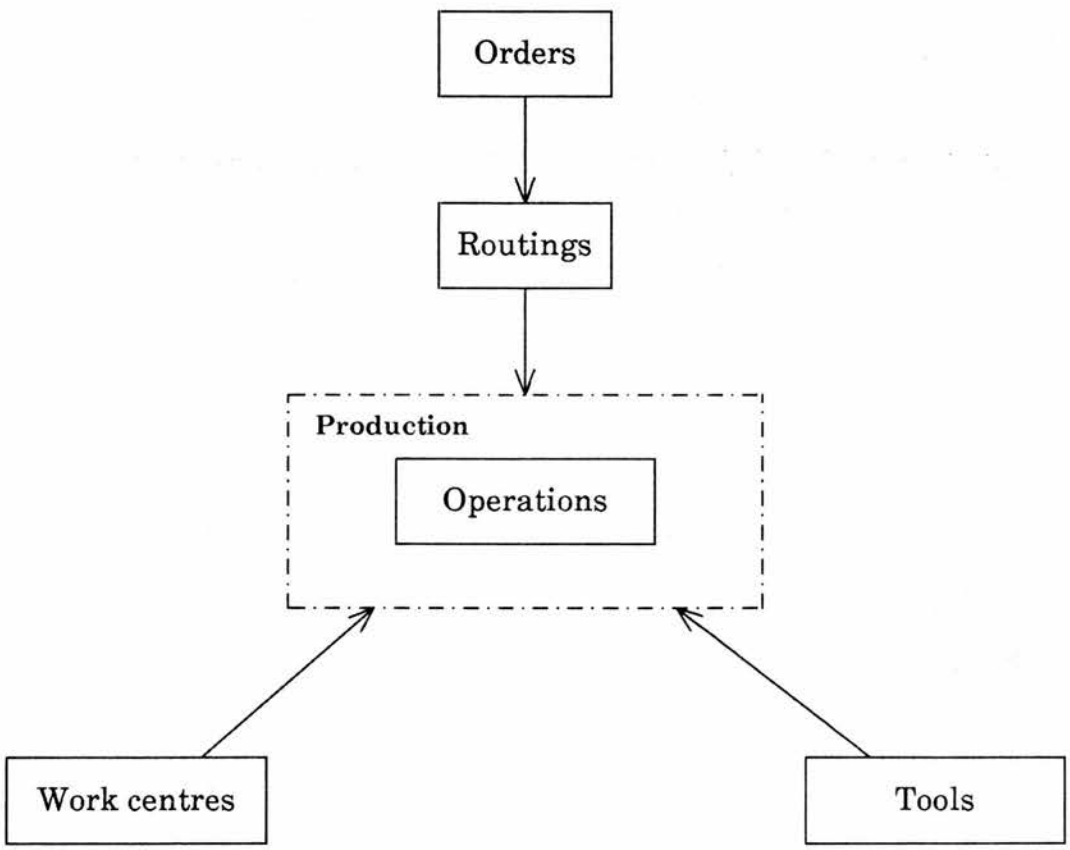


Figure 5.1 Organisation model

On a shop floor, the operations are performed and the final products are produced. Before any operation can be loaded, a machine has to be selected; tools required have to be there and ready; local constraints that are concerned with this activity also have to be considered. All these factors need to be represented in the shop model. In RESS-II, a hierarchical structure is used to express them and their relationships with each other. The process routing of a particular part (or an order) is defined as a set of manufacturing operations, and a sequence of these operations describes the precedence (constraint). Each operation is represented with machine requirements, tool requirements and manufacturing requirements. Individual components are linked with their entities, such as a group of identical machines composing a work centre. Figure 5.2 illustrates an example.

In the following, we will present how each scheduling concern is represented in the RESS-II scheduling model.

(1) Routing

A routing is represented in RESS-II by a part number which denotes a particular product and its manufacturing operations. There is an initial routing data file which stores all the routings of the parts that the factory can produce.

[Part-Number:
    [operation 1]
    [operation 2]
    .
    .
]

(2) Operation

An operation is one of the steps through which a part is manufactured. It is represented by its requirements (machine, tool, running time and set-up time), alternatives and preference constraints. (The operations of each part are stored in the routing file under its part number.)

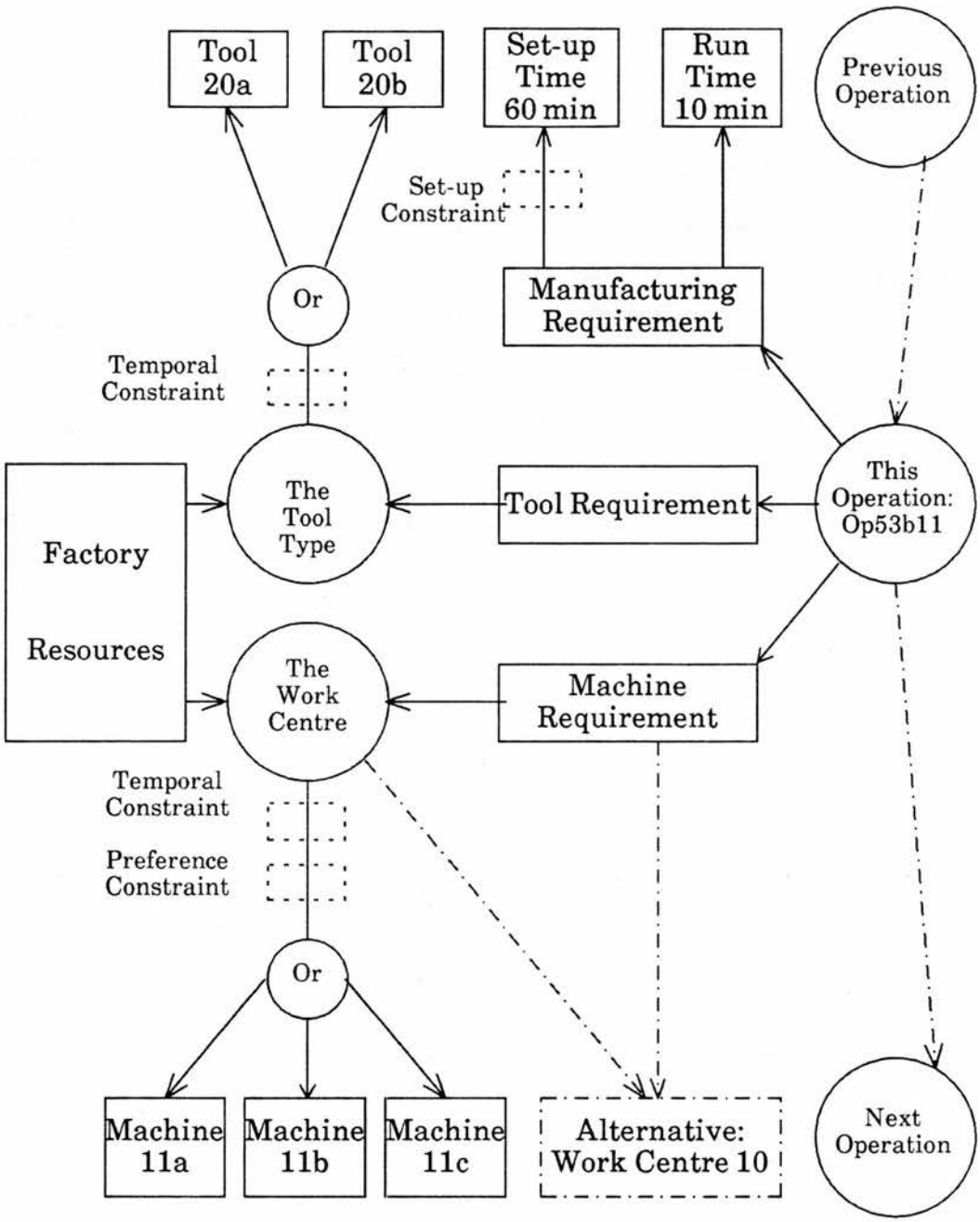


Figure 5.2 Operation-resources relationship model with attached constraints: a sample

[Operation-1
    [Work-centre:
        Alternative-work-centre:]
    Tool-required:
    Run-time:
    Set-up-time:
    Preference-constraints:
]

Alternative work centre above denotes the work centre on which the operation can also be performed apart from its original work centre. Normally, there are two ways to get an alternative work centre for a given operation. One is to ask for this information from the production engineer (or the production engineer will specify this initially during process planning). The other is to reason with the geometric features of the part to be manufactured, the technical requirements of this operation, and the functions and various limitations of all the work centres to try to find an alternative work centre. Generally, the technical issues can be considered very easily in a system, but the geometric shape of a part is very difficult to be represented and reasoned with. This is a research area in its own right and is not the aim of this project. But, in some simple situations, such as those considered by ISIS, only a few specific parts with similar geometric features are produced in a factory. It is much easier to represent and reason about them. In RESS-II, we suppose that the alternative work centre for a given operation is specified by production engineer, so it is also stored in the routing description of a part.

For the preference constraint, i.e. machine preference, the situation is the same. It is normally the shop foreman's desire to load an operation on one machine instead of another. His reason for doing so can vary a great deal. For example, the machines in a work centre, although appearing to be identical, do have some differences: the accuracy could be different, the operators could have different skills. Although these attributes could be stored in a knowledge base of a system for deciding the preferred machine for a particular operation, they can vary from

situation to situation. So, in RESS-II, we also assume that this information will be given by management and hence is also kept in the routing description.

(3) Work centre

A work centre is a group of physically identical machines. It is represented by a work centre number and the number of machines it has. In RESS-II, this information is stored in a work centre file.

[Work-centre-number:
Number-of-machines:]

An example layout of work centres and machines is shown in Figure 5.3.

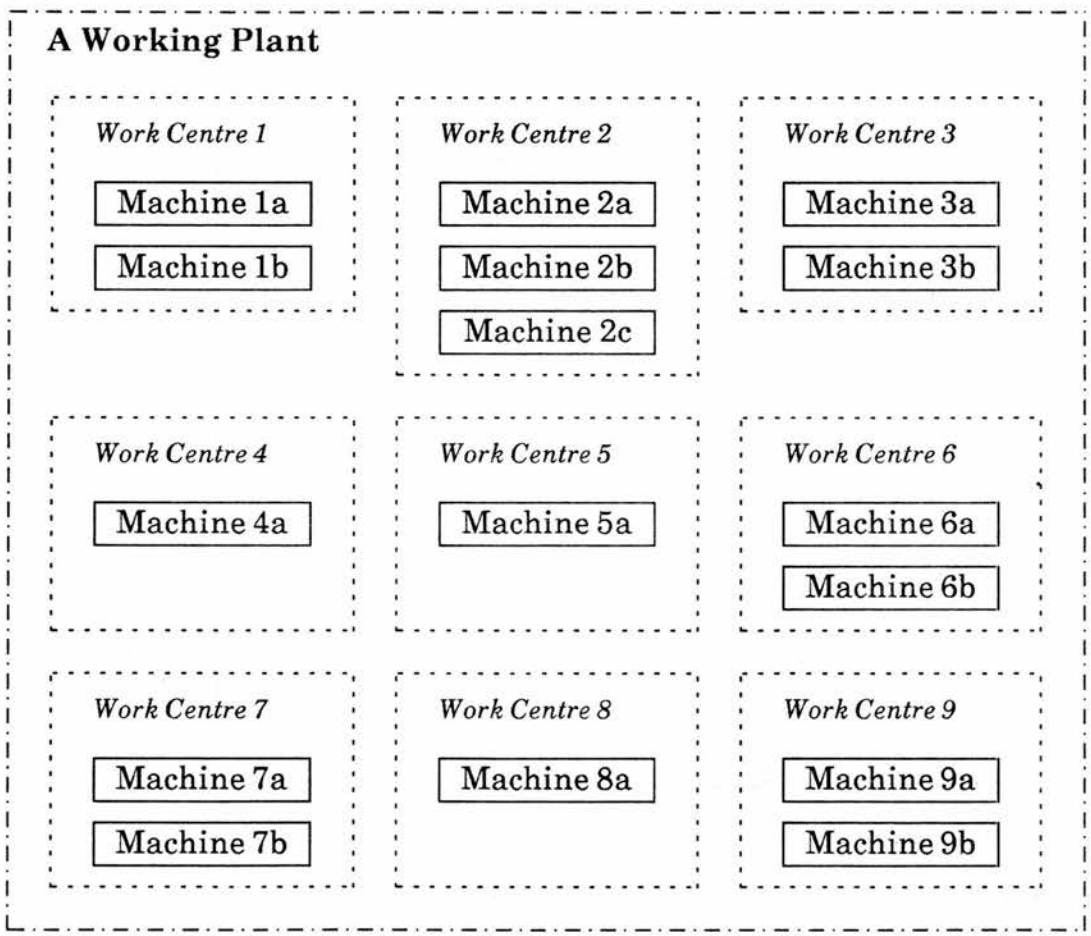


Figure 5.3 Work centres and machines in a working plant

(4) Tool resource

It is described in the same way as the work centre

(5) Order

Any type of orders, forecast orders, customer orders, stock orders, etc., can be created. They are described with order number, part number, quantity required, Proposed start time, delivery due date and management assigned importance value.

[Order-number
Part-number:
Quantity-required:
Proposed-start-date:
Due-date:
Importance-value:
]

Part-number denotes the specific part which an order requires. The proposed start time denotes when the order can be released to the factory shop floor for production, which is normally decided by management according to pre-production preparations and work-in-process time. The importance value denotes the relative importance of an order compared with the other orders and is also assigned by management. It ranges from 1 to 10. The bigger the number, the more important the order is.

Since RESS-II is a reinforcement scheduling system, the scheduling is different at different stages. Hence, the factory model required for different stages will also be quite different (this does not mean changed models but rather different levels of detail). At the first reinforcement building stage, only an aggregate level of the factory model will be necessary for its function. Some details are not considered, such as local constraints. Figure 5.3 shows the model for this stage. But in monitoring and rescheduling, more things should be modelled as unexpected problems will arise in a real manufacturing

process. This will be discussed in chapter 7 entitled "Shop Floor Monitoring and rescheduling".

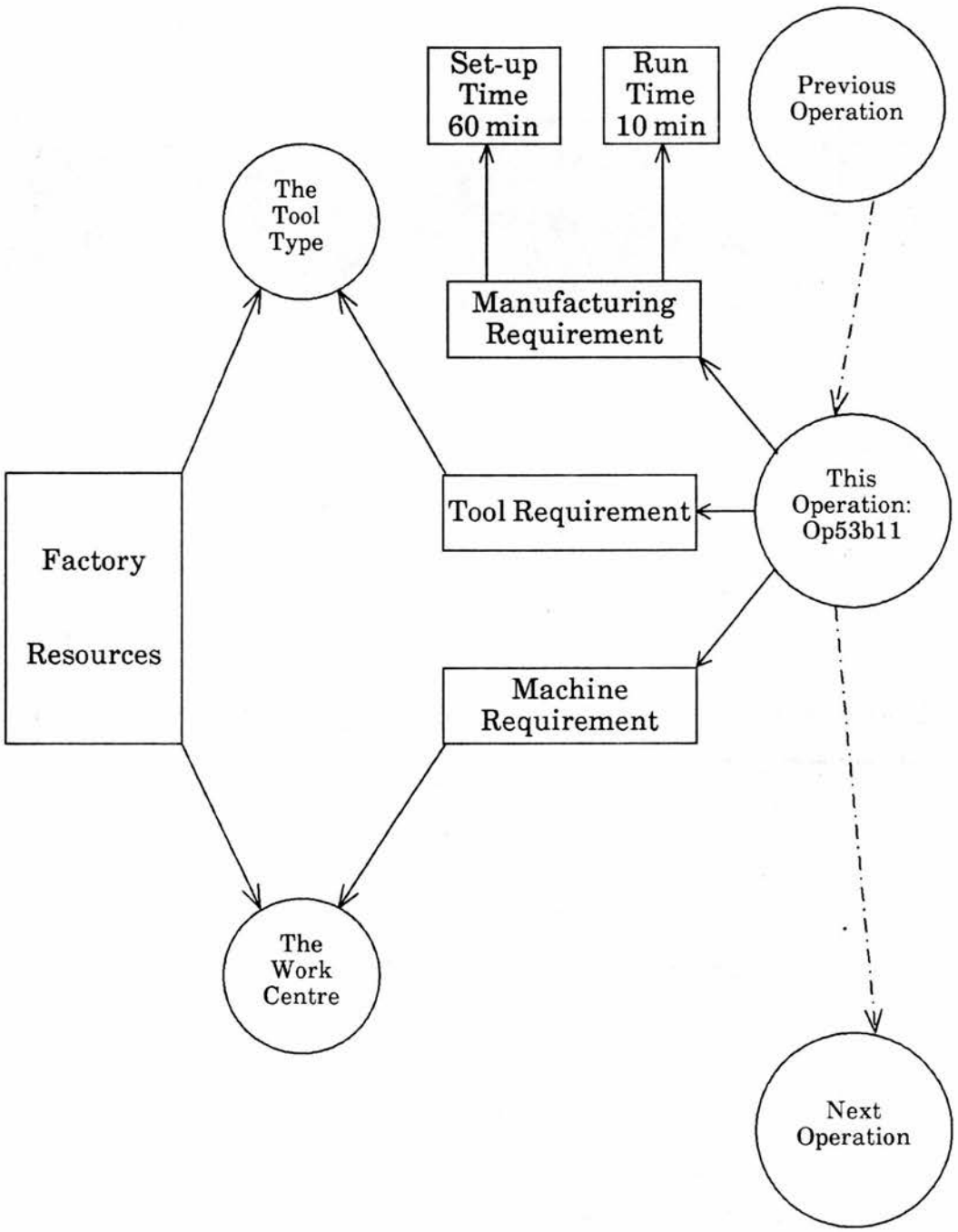


Figure 5.4 Operation-resources relationship model for reinforcement stage: a sample

5.2 Schedule Representation

After the factory has been modelled and shop concerns have been represented, the next thing to do is to represent a schedule. Such a representation will also capture the intent of the scheduling process. In other words, this representation will not only form the basic structure for a schedule but also indicates how scheduling will be performed.

In RESS-II, scheduling is defined as loading manufacturing operations to their machines. So, to schedule an operation is actually to put this operation onto its machine to occupy a period of time. Unlike many planning systems, this process is not a resource reservation process. The reason is that in each scheduling cycle, the scheduling (or planning) action starts from selecting a machine on which to schedule and then finding an appropriate operation to be scheduled on it, rather than laying down operations and correcting interactions first and then checking the resource availability as in most planners. However, tool scheduling is a process of resource reservation since its availability is only checked against the current situation (or state) of the selected machine and the ready operations.

In the light of this scheduling definition, a schedule in RESS-II is represented as machine schedule (or plan) or machine loading diagram. It is very different from the plan representation of partially ordered action networks in non-linear planning, where actions and their orderings form the fundamental basis for the representation. In a machine schedule (or plan), however, it is the machine resource that builds up the structure of schedule representation. This schedule specifies how each machine is occupied by different operations. It also shows when a machine is in an idle time period (no operation being processed in this time period). Generally, this schedule shows the loading situation of each machine in different time periods. The tool schedule is different, although in the similar form as the machine schedule, being directly derived from the machine schedule.

For some purposes, in RESS-II a schedule is also expressed by an order schedule. But, this is just a copy of the machine schedule in the orders' routings, which specifies when

each operation should start and when it should finish in connection with the other operations in each order. The start time of the first operation in an order tells when this order starts to be manufactured on the shop floor. The finish time of its last operation tells when this order will be finished and whether it can be finished when due. The time intervals between two successive operations show the queuing time (or waiting time).

However, as far as the time assignment is concerned, these different schedules are essentially the same but stored in different forms in different places and with different purposes. Every operation's schedule in its order will find a corresponding schedule in the machine schedule (although it may not find one in the tool reservation because not all the operations need tools). But the underlying intentions of these schedules are not the same. The machine schedule is the most important one and hence provides the driving force, while the order schedule and tool reservation are only its derivatives.

(1) Machine schedule

As the resources in scheduling domain are shared, their capacities are described over time. Hence, machines and tools in RESS-II are represented as objects with fixed capacities during the time period concerned. In RESS-II, the machine schedule is expressed as follows, with a hierarchical structure being used.

```
[Work-centre-number:
        Machine-1-schedules:
        Machine-2-schedules:
        .
        .
        Machine-n-schedules:
]
```

A work-centre-number denotes a particular work centre. A machine-n-schedules represents the schedules on machine **n** of the work centre. The capacity of a machine is organised into weeks (that is to say that the schedule time of the machine is divided into weeks). The schedule time bounds of an operation are specified with the week and the time in the week. The reason

for using this, rather than a continuous time and putting a convenient starting point as the base, is that this approach matches that being used in industry [24] [43] and it is also easy for the system to access a certain piece of information. For example, if the system wants to find out an operation's schedule in the machine schedule, it can first go to the order schedule to find out which week it is in, and then go to that week in the machine schedule to find it. This will involve less operations' schedules being checked than with continuous time. Otherwise, the system may have to check all the schedules of operations right from the beginning in order to find a simple piece of information.

Machine-n-schedules is represented like this:

```
[Machine-n-schedules:
        [Week-number: 1
            Operation schedule-1:
            Operation schedule-2:
            .
            .
            .
            Operation-schedule-n
        ]
        ...
        [Week-number: n
        ...
        ]
]
```

The schedule for each operation in the machine schedule is like this:

```
[ Order-number:
    Operation-number:
    Operation-start-time:
    Operation-end-time:
]
```

Machines are not occupied by operations all the time. Typically, there are idle time periods. These idle time periods in RESS-II are also made explicit in the

machine schedule and represented as special operations. Obviously, they will not be represented in the order schedule.

[Idle-time
    Idle-start-time:
    Idle-end-time:
]

Graphically, the machine schedule can be represented as in Figure 5.4. Each machine and its capacity is expressed as a continuous dashed line. The length of the line represents time. Operations scheduled onto each single machine are denoted by filled lines. The length of each these lines expresses how long an operation will be performed on a particular machine. The start and end point of this line denote operation start time and operation end time.

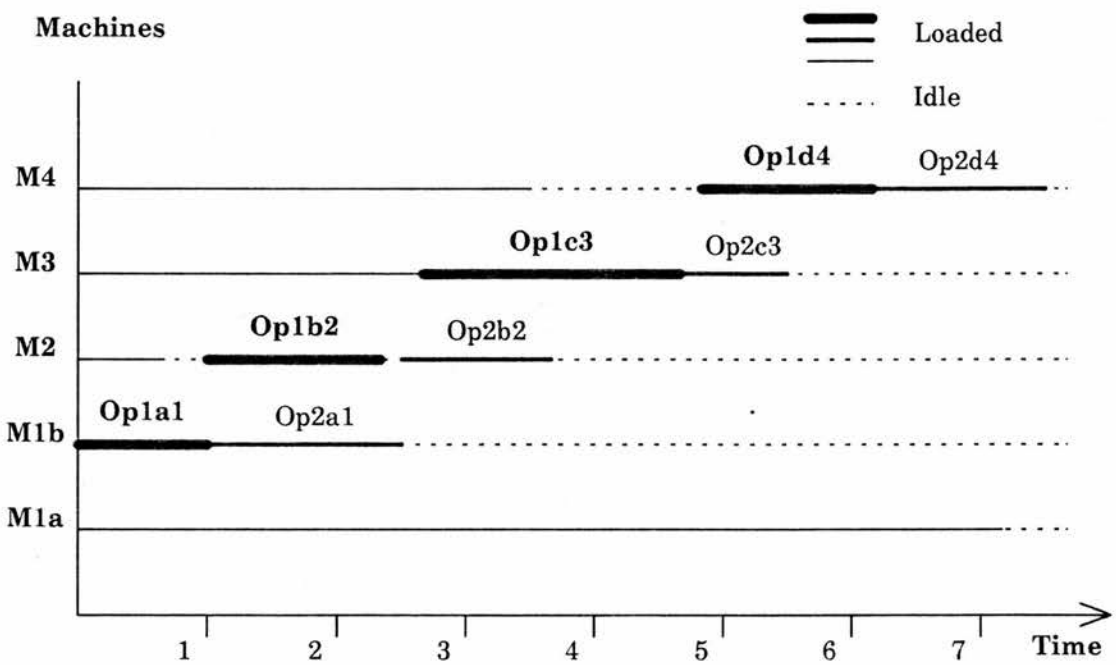


Figure 5.4 Schedule representation

(2) Tool reservation

Tool reservation has the same kind of representation as the machine schedule, but it is not treated the same way in the scheduling process as the machine schedule.

[Tool-type-number:
Tool-1-schedules:
Tool-2-schedules:
.
.
.
Tool-n-schedules:
]

Tool-n-schedules denotes the schedules on tool **n** in the same tool type and it is represented like this:

[Tool-n-schedules:
[Week-number: 1
Operation schedule-1:
Operation schedule-2:
.
.
.
Operation-schedule-n
]
...
[Week-number: n
...
]
]

Operation schedules and idle time periods are also represented here in the same way as in the machine schedule.

(3) Order schedule

This schedule is expressed by the schedules of operations. It is kept in the original routings specified by the orders, but with more information added to them. An operation schedule here is different from an operation schedule in the machine schedule and the tool reservation.

[Order-number: 1
Operation schedule-1:
Operation schedule-2:
.
.
.
Operation-schedule-n
]

Operation schedule is represented like the following. It is actually the same as the operation representation but with two more slots created to express the specific machine on which an operation is going to be performed, and the start time and the end time of the operation.

[Operation-number:
[Work-centre:
Alternative:
]
Tool-required:
Set-up-Time:
Run-Time:
Preference-constraints:
Machine-number:
Operation-start-time:
Operation-end-time:
]

5.3 Schedule Constraints

Constraint representation in RESS-II is not as sophisticated as in ISIS [31]. Local preference constraints which are concerned with each particular operations are expressed in the operation description. Constraints which are connected with orders are represented in an order description. The method of handling these constraints are represented with procedures. They describe how each constraint is applied and the situations in which they can be satisfied. There is no attempt to represent the constraint any further.

There are several issues about constraints that need to be addressed.

(1) Resource problem constraints

This is a new concept introduced in this research. As discussed in chapter 3 and 6, these constraints represent the resource problem situations, and how they can be predicted and resolved at a local scheduling state.

(2) Constraint interaction

Constraints in scheduling are not independent. Typically, they interact with each other. Chapter 4 discussed this problem.

(3) Constraint relaxation

Not all the constraints can be satisfied at a scheduling situation. When they cannot be satisfied, they are relaxed. This has also been discussed in chapter 4.

(4) Constraint relevance

Constraint relevance defines the situations under which a constraint should be applied. Not all the constraints will be considered all the time, and some

constraints will not participate in decision making in certain situations. In fact, there are two types of constraints. The first type of constraints are considered whenever a decision making is encountered. There is no relevance problem about these constraints. The second type of constraints do not need to be considered all the time as they depend on particular conditions or situations. For example, if the set of operations that is being evaluated is not urgent, then there will be no need to take importance and urgency constraints into account in the scheduling decision making.

Since a constraint decomposition approach is used in the RESS-II system, different stages will also emphasize different constraints' satisfaction.

(5) Constraint importance

Typically, there are many constraints that need to be considered at a scheduling state. The system evaluates each alternative choice according to the assessment of these constraint's satisfaction. In RESS-II, each constraint (including those resource problem constraints) is given an importance value. Satisfaction of this constraint will be expressed by this value, which is then incorporated into the existing rating of the operation being considered. There are two important points that need to be addressed.

(a) Different constraints may have different importance values, since not all the constraints are equally important to scheduling. Some constraints are more important than others (resource problem constraints are likely to be more important than an operation preference constraint).

(b) The importance value of a constraint also varies over different situations. A constraint may be relevant to many situations, but could play different roles. In some situations, it could be very important to the decision making but in the other situations, it may not appear to be so critical. For instance, when there is no urgent operation, although the urgency of the set of candidate operations varies a great deal, these urgency and order importance values will not be so important. The other constraints

will play more important roles. To express this in RESS-II, different parameters are used to change the original importance value of the constraint according to different scheduling situations.

(6) Constraint generation

Many constraints are introduced dynamically as the construction of the schedule proceeds. Some constraints may also be removed. For instance, if the system finds that the work centre currently being considered is a bottleneck work centre, it will try to find alternatives for the operations. If the alternative work centres are found and some operations have been scheduled on the machines in these alternative work centres, a resource availability constraint will be introduced to the rest of the operations as the operations being scheduled on the alternatives may reserve the tools that these operations (which have to be scheduled in their original work centre) need. In this case, the system will need to check the resource availability for the rest of the operations again before it can try to schedule one of them onto the original machine. At the same time, scheduling of certain operations on the machines of alternative work centres may ease the load in the original work centre so that this resource problem constraint may be removed. Dynamic creation of constraints is accomplished by attaching a generator to the constraint satisfaction procedure. The removal of certain constraints is also represented in this way, but it requires constant checking.

(7) Constraint substitution

As mentioned before, the global constraints are difficult to consider at local scheduling states, therefore, in RESS-II, they are substituted by some other constraints that can be considered locally, such as using order importance and urgency constraints to express the due date constraint.

(a) Order relative importance: This constraint is different from the constraint importance discussed above. It expresses a management assigned value

of how critical an order is compared with the other orders, or how important it is to satisfy the due date constraint of the order.

(b) Order urgency: This constraint expresses a priority value of the order being considered. Its value is produced dynamically in the scheduling process, which means that the urgency is not a fixed value for each order. After some operations in an order have been scheduled, the calculation of urgency of the order only considers the present operation and the ones after the present. The operations before the current operation will not be taken into account. Although it is for the scheduling of this particular operation, this urgency is the urgency of the order at this moment.

This completes the review of the model of the factory environment. The next chapter will show how this model is incorporated into the RESS-II scheduling system.

Chapter 6

The RESS-II Scheduling System

RESS-II is the second version of the RESS job shop scheduling system. Its scheduling environment has been described in chapter 2. The central approaches it uses are reinforcement scheduling and constraint decomposition. It has, in total, five stages of scheduling, grouped into four parts. The first part is the reinforcement building part, which has two scheduling or planning stages. The second part is the schedule construction (or the main scheduling part), which has only one stage of scheduling. The third part is the schedule refinement part, which has also only one stage of scheduling. The fourth part is the monitoring and rescheduling (or reactive scheduling), which also has only one stage. Each of these stages represents a complete scheduling process and can produce a complete schedule. The final schedule will be produced from the refinement stage. The schedules produced in the other stages are just reinforcement and intermediate schedules for the later stages.

In the reinforcement building part, the first stage builds a capacity plan to show the possible conflict situations between machine capacities required and machine capacities available in order to meet the due date constraint of each order. The second stage builds

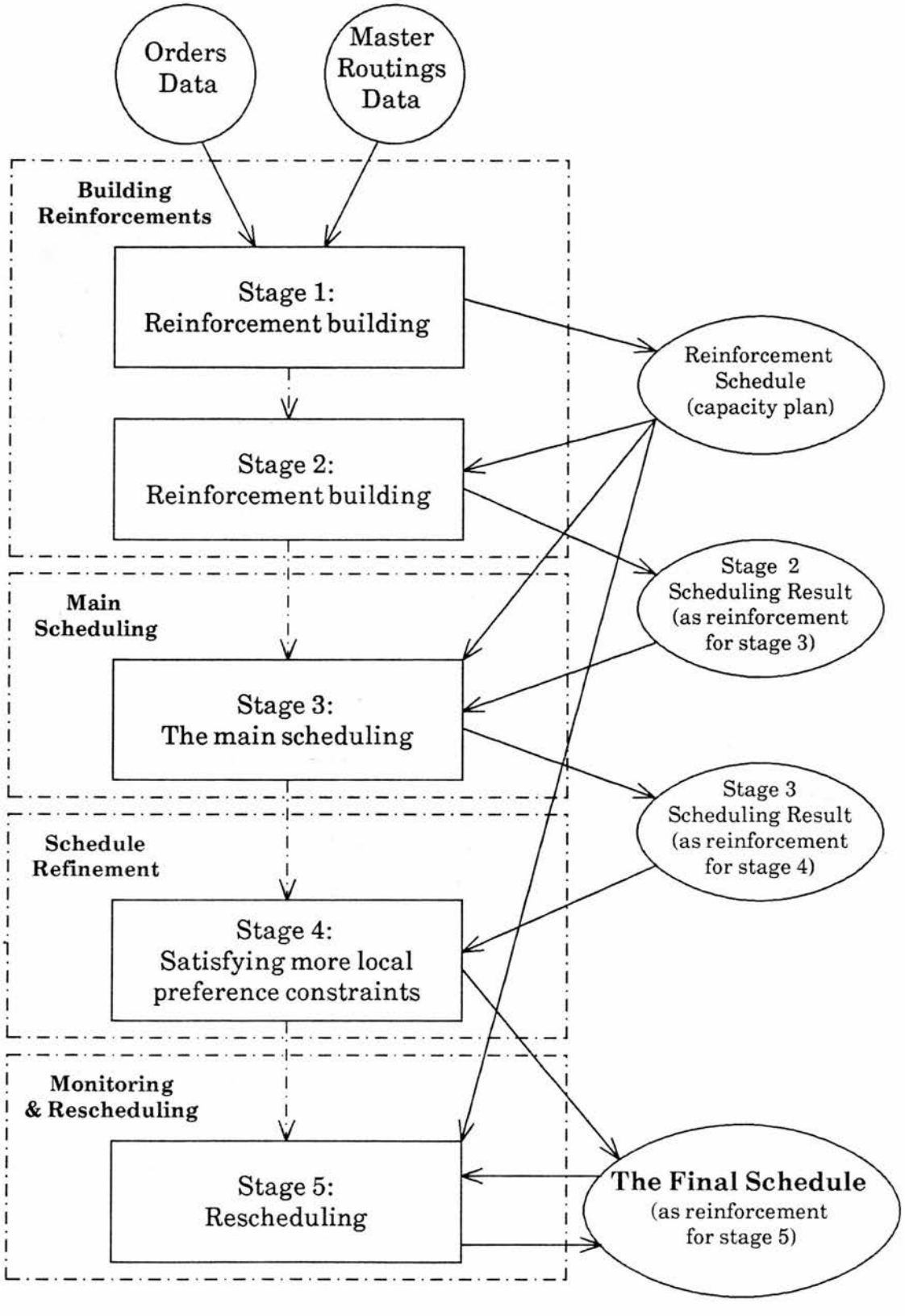


Figure 6.1 The RESS-II system structure

a schedule to expose the conflict situations between the due date constraint and the work-in-process constraint. The main idea of resource allocation and the implementation of the reinforcement approach in RESS-II are discussed here with the description of this stage. Both of these schedules will act as reinforcements for the decision making in the construction of the schedule. In schedule construction, the system generates a schedule by using the problems manifested in those two reinforcement schedules. The schedule refinement part refines the schedule constructed in order to satisfy more local preference constraints, and the rescheduling part deals with the dynamic nature of the job shop and will be discussed in the next chapter.

Figure 6.1 illustrates the system structure of RESS-II. The dashed lines with arrow heads indicate that one stage is before or after other stages.

6.1 Reinforcement Building

6.1.1 Building and Analysing the Capacity Plan

The manufacturing process is directed towards meeting the objectives defined by management (such objectives are different from the objectives set for scheduling). The manufacturing objectives specify what products are to be produced, in what quantity, by what date, etc. The source of these data can be customer orders, forecast of future demand, or a combination of both. These are non-engineering sources. Their requirements are not concerned with actual plant capacity. It is the scheduling task that incorporates these requirements into the real situation of production plan and shop capacity, i.e. to plan the shop activities in order to meet these demands according to the actual resource available in the plant. In order to construct a good schedule to meet these objectives according to the actual plant capacity, the impact of these objectives on plant capacity and load balance should be assessed. In other words, the capacity

required and the capacity available should be evaluated first before performing any scheduling. This is the aim of this reinforcement building stage.

This stage will build a capacity plan to show the problems between the resource capacity required and resource capacity available in order to provide the schedule construction process with an overall picture of the forthcoming events (i.e. the reinforcement plan). As discussed in chapter 3, this process will not consider a lot of detailed information, so it is not a schedule that can be used directly in the real factory.

According to the discussion of building reinforcement plan in chapter 3, an important resource and an important constraint should be chosen first: and clearly these should be the machine and the due date constraint respectively. In chapter 3, we also mentioned that there can be many ways to build a reinforcement plan. Here, there are two obvious ways. One is to schedule according to the availability and the finite capacities of work centres, while leaving the scheduling objectives described as objective constraints, such as due date, unattended. The post-level analysis will be done globally to find the bottleneck work centre(s) and the possible jobs to be delayed. The other is to assume that every order will be finished on time (or to make the assumption that all the required capacity will be available), hence building a schedule based on work centres without considering the finite capacities of them. This method leaves the work centre capacities unattended and consequently shows the loading condition.

In RESS-II, the latter method of building the reinforcement schedule is used. This does not imply that this method is better than the first. We plan to include the first method as an option at some point in the future.

As this scheduling process disregards the finite capacity of the resources, the infinite scheduling technique is used. In this technique, the operations' loads are put in the different time periods as specified by the lead times of the operations. The basic resource unit is the work centre rather than individual machines. During the building process RESS-II also records the usage of other less important resources. The scheduling method

is single order scheduling (it can also be done by using parallel scheduling but it will be more complicated).

In order to carry out this reinforcement scheduling, the system requires the following information.

(1) Due date of each order

Infinite scheduling satisfies the due date constraint of each order. This information decides in which time period the last operation of an order should be loaded. It also helps to decide the lead time for each operation.

(2) Lead time for each operation

This is a time period within which an operation is supposed to start and finish. It consists of set-up time, running time and queue times before and after processing.

(3) Start time of each order

This time decides when the first operation of an order can be loaded to its work centre. In the situation that the lead time of each operation is not given this information and the due date information will help to decide an approximate lead time for each operation in an order.

The due date of an order is given according to the customer's requirement, while start time is decided by considering the preparation time, material handling and work-in-process time. In other words, they are all decided by management. Lead time for an operation is sometimes also provided by management. Different companies have their own value for lead time according to their own situations and their experience [24] [43]. But, normally the lead time specified by a company is not concerned with due date of a particular order at all. Instead, it is a fixed time window in which the operation

should be performed. If the due date constraint cannot be met by using the lead time provided for each operation of an order they will believe that this particular order is unlikely to be finished on time. This concept is slightly different from that of RESS-II.

As the reinforcement scheduling or planning process concerns the scheduling of the orders and determination of the capacities required to ensure that the due date constraint will be satisfied, so, the criterion for choosing lead times is to ensure each order is finished on time if they start at the requested start time. We argue that although the set-up time and running time of an operation can be decided, there is no way to decide the waiting time of an operation as this is very much context dependent. To put a fixed lead time for each operation will be inadequate for reinforcement scheduling. Instead, we consider the lead time of each operation in an order along with start time and the due time of the order and ensure that the due date will be satisfied. Two simple rules are used for this purpose: equal waiting time on each work centre for each operation or equal lead time for each operation of an order. If the company provides the lead times, they will be used if the due date constraint is not violated. Otherwise, they will be shortened accordingly.

— Equal lead time rule: supposing the start time of an order is ST and the due time is DT; this order has N operations. Then the lead time LT, for each operation will be:

$$LT = (DT - ST) / N$$

— Equal waiting time rule: Total operations time of the order is TOT. Operation time of an operation is OT. Waiting time for each operation is WT. The lead time for each operation is given by:

$$WT = (DT - ST - TOT) / N$$

$$LT = WT + OT$$

Both of these apply to the RESS-II system. This scheduling process basically consists of two steps: preparing input for scheduling and infinite scheduling. It is a cyclic process

going through all the orders (not prioritised) needed to be scheduled one by one to arrive at a complete rough plan as the overall picture for the later detailed scheduling process.

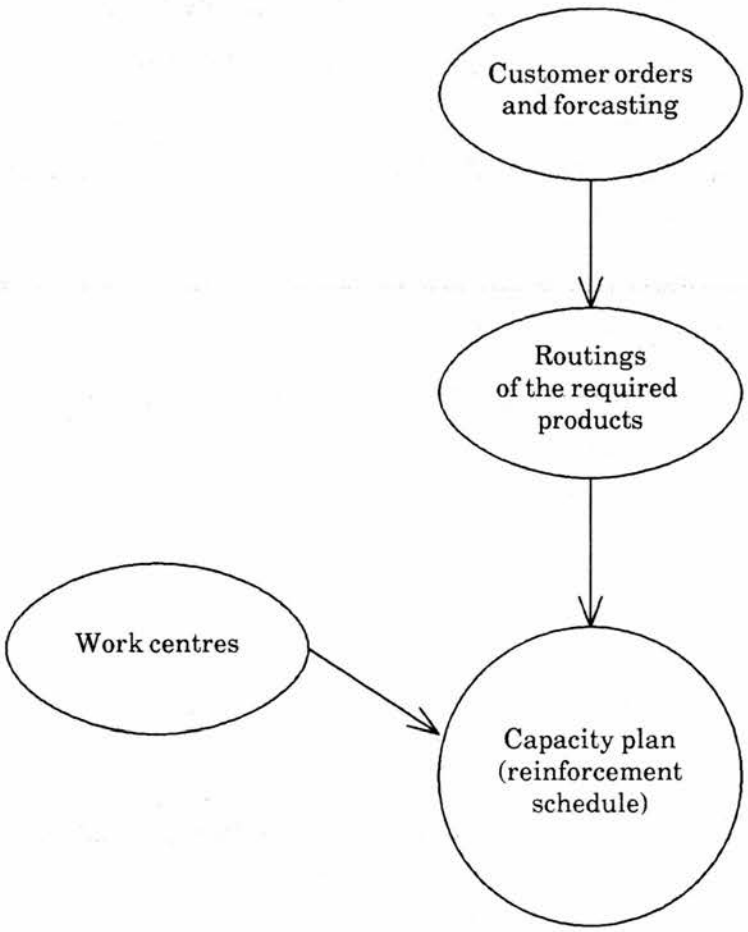


Figure 6.2 Capacity planning data and result

The initial data file for scheduling of RESS-II includes the order file, the routing file and the work centre file. (The order file contains the descriptions of all the orders that need to be scheduled, and each order is expressed by order number, quantity, due date, start date and managment assigned importance value. The routing file contains all the routings of the products that the factory can produce. Each of these routings gives a list of operations from which the product can be produced and each operation is described by a work centre on which its processing should take place, the sequence of operations, lead

time, set-up time and standard machine time (running time). The work centre file details work centres available to the plant and how many machines each work centre has.) By means of their information, we can explore each product in the order file and accumulate the workload at each work centre by time periods.

(1) Preparing input for scheduling

The system first extracts an order from the order file, and then retrieves the routing of the product this order requires. Then, according to this routing and the information provided in the order description, each operation's loading time is computed by using running time, set-up time and quantity required. After that, the system calculates the lead time (if it is not provided) for each operation in the order by applying an equal waiting time rule or equal lead time rule, and using due date and requested start date. If the lead time is provided in the routing data of the products, it will be used.

(2) Infinite scheduling

As the lead time for each operation has been established, this step loads the operation onto its work centre in terms of operation loading time (set-up time plus running time). The system first retrieves the work centre that the operation requires and then adds the load of this operation to the time periods specified by lead time bounds (the earliest start time and latest finish time). If the lead time bounds cross two periods, the load will be split proportionally and added to both of these time periods. All the operation will be loaded in this way.

Here, the plan (or the reinforcement schedule) is represented with three pieces of information: one for work centres: one for time periods and one for capacities required in these time periods. Currently, RESS-II uses a weekly basis for its time period. The length of this time period can be changed according to different situations.

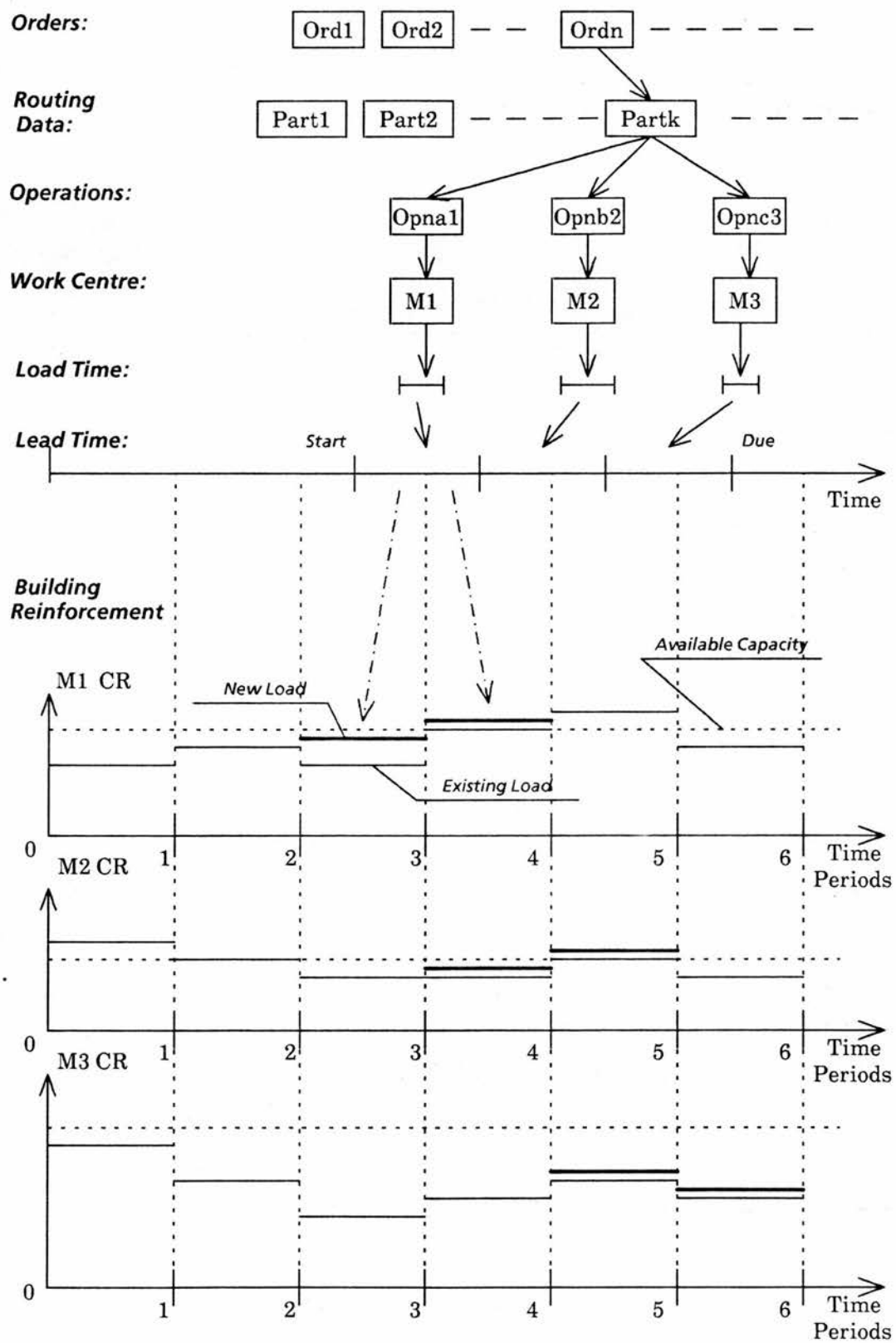


Figure 6.3 Building a reinforcement plan *(CR: Capacity Required)*

After this process finishes, the system goes back to (1) and another order will be retrieved. As this cycling progresses, all the orders will be loaded. Figure 6.3 shows this process.

An example work centre load in different periods is shown in Figure 6.4, which illustrates the profile of the loading conditions of this work centre. Typically, from these profiles, one can get a lot of information. In terms of all the work centres, we can learn which work centres in the shop are overloaded (global bottleneck work centres) and which are underloaded, while in terms of each individual work centre, we can learn whether the overloading occurs in all time periods or there is a mixture of overloaded (bottleneck periods) and underloaded periods and what the average load is.

Obviously, this capacity plan will not be realised on the shop floor, but it is a good starting point for a later schedule. This plan or schedule represents a framework for the prediction of overall plant performance. Through analysing this plan, the second stage will try to balance the load to avoid the overloaded periods from happening or to reduce their effects on the schedule.

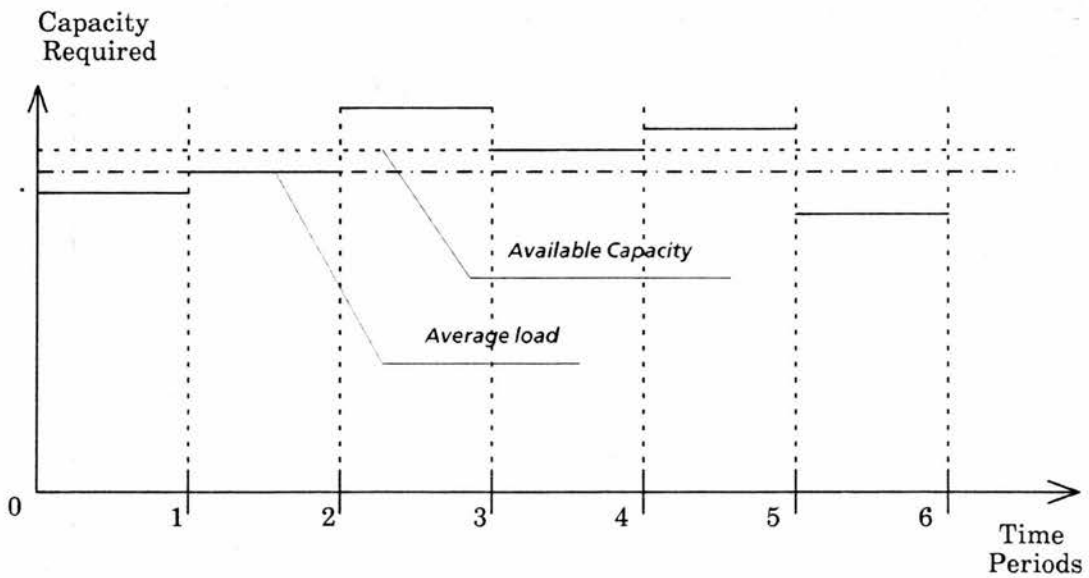


Figure 6.4 A sample machine centre load profile from the reinforcement schedule

After the reinforcement scheduling has been finished, that is all the orders have been scheduled, the capacity plan analysis is performed. It analyses the schedule result to find out global resource problems, i.e. global bottleneck work centres. Although the local bottleneck periods on different work centre can also be detected, they are not reliable as scheduling is context dependent. It will be much more effective to detect the local bottlenecks during the schedule construction process.

This analysis can also be applied over a very broad area to provide information to management. According to this information, management may change the shop situation in order to solve some problems, e.g. by increasing capacity or subcontracting, etc.

6.1.2 Reinforcement Building: Stage two

After the capacity plan has been generated and the analysis has been done, the system is ready to build another reinforcement schedule, which takes the capacity plan as its reinforcement, in order to satisfy the due date constraint. The resulting schedule will also show the conflict situations between the due date constraint and the work-in-process constraint, which in fact it is the real purpose of this stage.

At this stage, those less important resources such as tools and less important constraints such as machine preference and set-up time will be taken into account. The scheduling process is a single-pass process which means that the scheduling goes chronologically forward and after any operation has been scheduled it will never be changed. The scheduling method used is parallel scheduling. The basic resource unit is the individual machine rather than the work centre.

The reinforcement approach to the factory scheduling problem has been predicated on the assumption that the key to successful schedules lies in appropriate allocation of resources to satisfy the constraints in the shop environment. Stated another way, in scheduling it is the resource capacity that restricts constraints' satisfaction, hence,

satisfaction of constraints can be interpreted as solving the resource problem. The system assumes that if the resource problems can be tackled the constraints will be best satisfied. It is an indirect way of reaching a good constraint satisfaction result.

So, the focus of decision making or choice making in scheduling is on predicting possible resource problems and trying to avoid them or reduce the effect of them. It has two sub-issues. The first one is the local evaluation of possible resource allocation, which includes the current loading analysis and local consideration of constraints of each operation and the shop. The second concerns the global effects of such possible resource allocation. It is an ability to differentiate amongst the alternative sets of scheduling decisions based on the prediction of the future impacts of such decisions. A good schedule requires the evaluation of these alternatives, which attends both to various objective concerns (constraints) and preferential concerns (constraints). In this section we present the second reinforcement building stage in RESS-II, which deals with these issues.

There are four steps in this scheduling process. The first one can be seen as the first level of choice generation and the second the decision making. The third one performs the scheduling action and the fourth one updates the reinforcement schedule. At each scheduling state (or in each scheduling cycle) a single machine is chosen and an operation is selected to be scheduled on this machine. While the process goes on, all the operations from all the orders will be scheduled.

(1) Finding the ready machine and the operations

(a) Finding the ready machine to set up a scheduling state

This step initializes the scheduling process by analysing the current scheduling situation to determine which machine to schedule next. As the parallel scheduling method schedules the operations chronologically forward, the system first selects a machine (a single machine) to schedule. In this case the machine whose last operation completed the earliest (the machine whose current load ends the earliest) will be chosen. Associating this machine with

its idle start time produces a scheduling state. Figure 6.5 (A) shows this process, in which M2a (machine "a" in work centre 2) is selected.

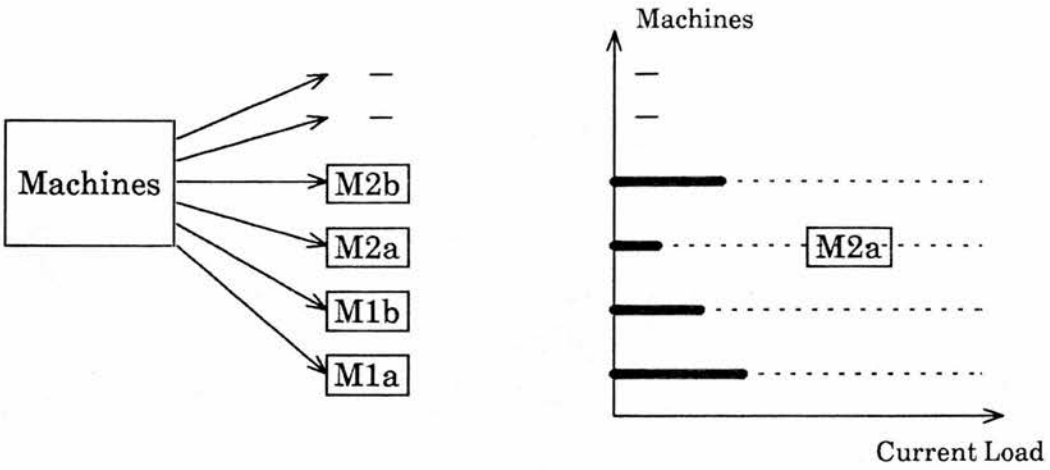
(b) Finding the ready operations waiting at this machine

According to the temporal constraint of this scheduling state (or the machine available time) and precedence constraints of each operation specified in the process routing description of each order, the system finds out the ready operations. Those operations whose predecessors have been completed at this state will be chosen. The resulting set of operations will be further scrutinized by the tool availability constraint (which checks whether the tools required by these operations are available at the moment) and in turn produces a subset of schedulable operations. This process is illustrated in Figure 6.5 (B), (the operations waiting to be scheduled at the top go through two constraints to produce a schedulable subset to be scheduled on M2a).

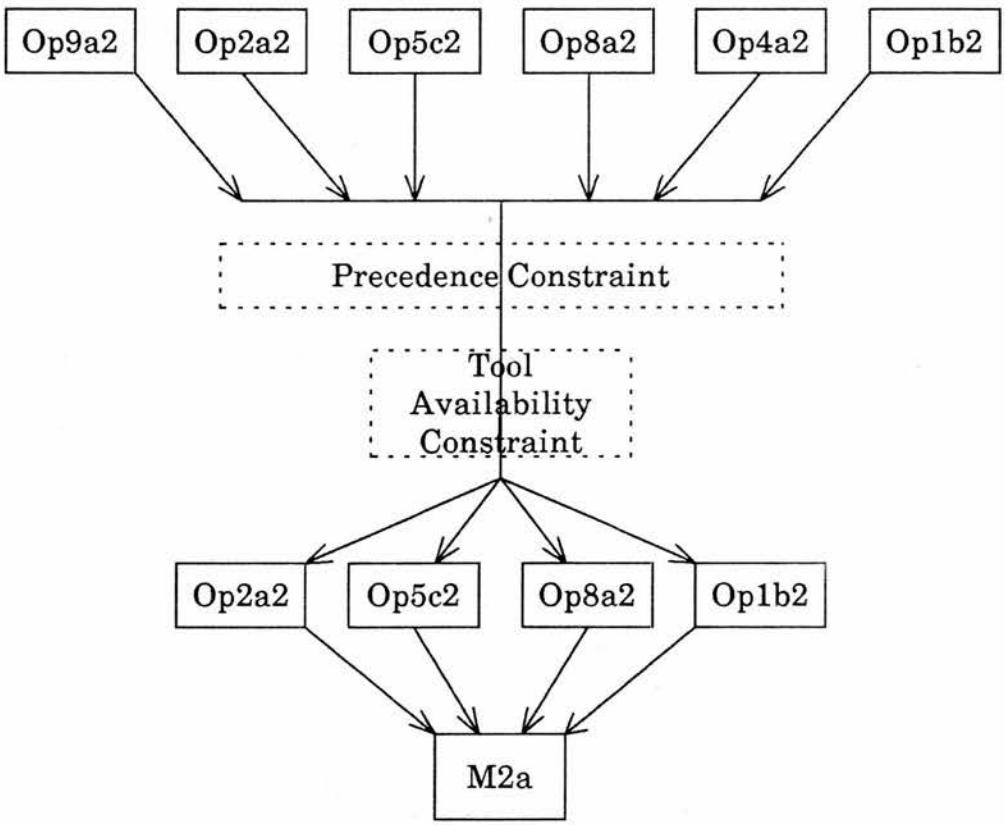
(2) Scheduling decision making

In any situation, it is almost always the heavily loaded work centres, both global and local (bottleneck periods), that influence the schedules most. So, dealing with these bottleneck work centres becomes an essential task in scheduling decision making. The substep (a) below deals with the bottlenecks directly by finding alternative work centres to reduce the load on the problem work centres. The substep (d) indirectly deals with the bottlenecks and other resource problems by properly sequencing the operations to be scheduled on each machine to balance or level the load to avoid or reduce the effects of these problems. The evaluation mechanism is based on a rating system, within which different constraints or problems are weighted differently.

Figure 6.6 shows a sample process. The operations (Op2a2, Op5c2, Op8a2, Op1b2) from the previous step go through all the substeps here and a single operation is chosen at last to be scheduled on machine M2a.



(A) Machine selection



(B) Finding the ready operations

Figure 6.5 Set up a scheduling state and find out the ready operations

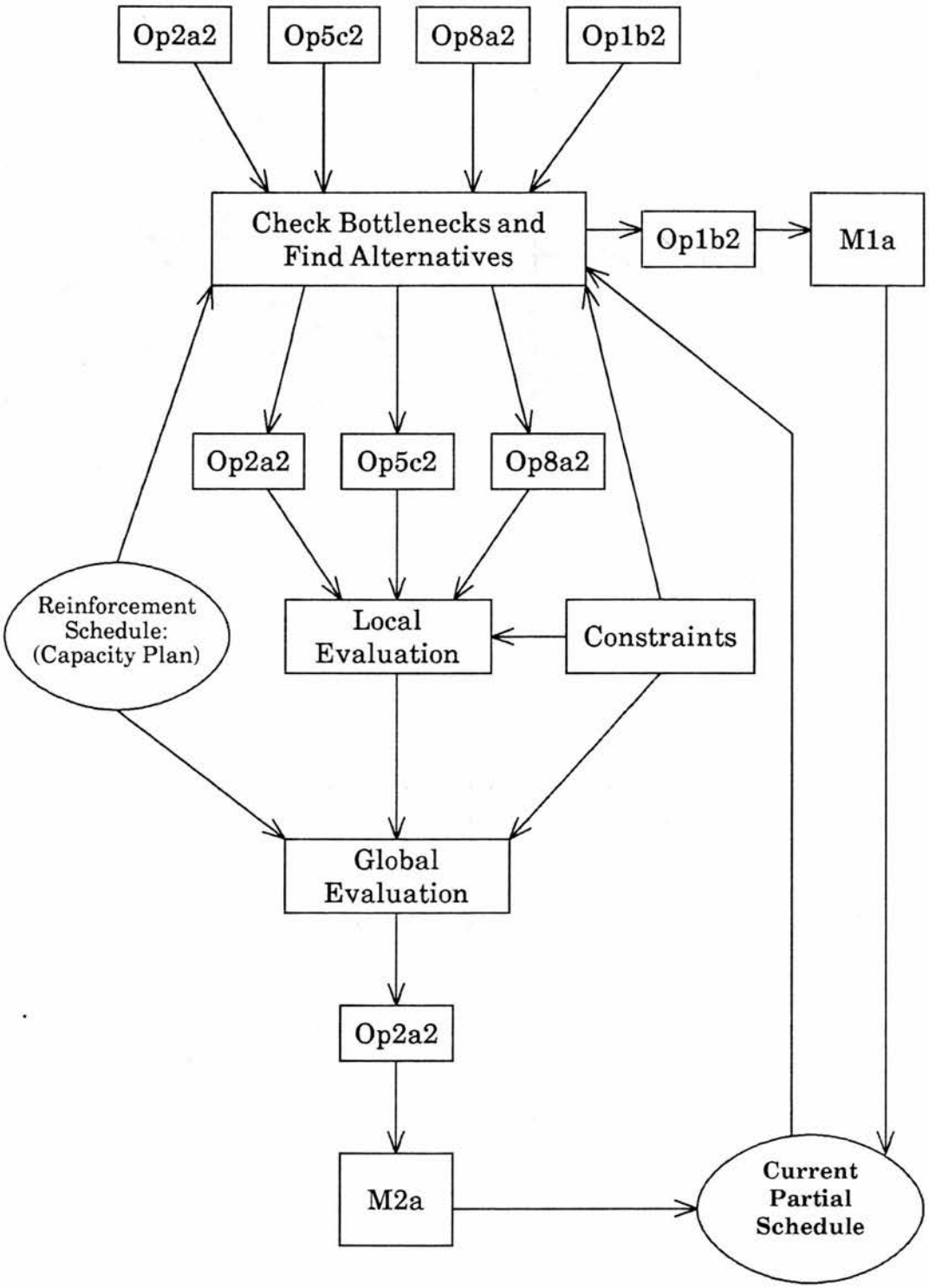


Figure 6.6 Scheduling decision making

Note that the substep "Check bottleneck and find alternative" scheduled operation Op1b2 on a machine (M1a) in the alternative work centre.

(a) Check the bottleneck and find alternatives.

In addition to the global bottleneck work centres, many locally highly-contentious work centres (local bottlenecks) may also exist and contribute to the scheduling problem. Many of the specific relationships and problems may emerge only after some amount of scheduling has been performed. As scheduling goes on, the load of some local bottleneck periods may be leveled or balanced and new ones may show up. Even the global bottleneck work centres may cease to be bottlenecks any more because of performing some operations on alternative machines. This substep first dynamically finds out if the work centre that the chosen machine belongs to is currently in a bottleneck period by current situation analysis, reinforcement plan analysis. If the answer is yes, then, the system will try to tackle it directly by reducing the load on it, i.e. trying to find out alternative machines. The current situation analysis gives the current load on the work centre which is to assess how much capacity has been used up till now. The reinforcement schedule analysis provides an expected load in the near future. Comparing these two loads with the capacity available, the system finds out if the work centre is in a bottleneck period or not. If it is, the system tries to find alternative machines in other work centres for this subset of operations coming from the last step.

The process of finding an alternative machine is similar to the scheduling process. It has two objectives; (1) ease the load on the original work centre by scheduling the operation(s) onto other work centres, and (2) avoid adversely overloading the alternative work centres. For example, it is no good easing the load of one work centre by putting some operations onto the alternative work centres and causing them to be overloaded which may in turn delay some orders. The process is like this.

— The system first picks up an operation from this set and checks to see whether there is an alternative work centre which this operation can be put in (as mentioned before in chapter 5 this information is stored in

the operation description). If there is one, then it goes to the next step. Otherwise, the operation will be left waiting to be scheduled in its original work centre.

— The system brings the operation to its alternative work centre and checks to see whether the loading of this operation will overload the alternative work centre. If this is the case, the operation cannot be put in this work centre. Such a checking process involves reinforcement plan analysis and the current partial schedule analysis on the alternative work centre. If the schedule of the operation will not overload the alternative work centre, the system pursues the process further in order to load this operation on a machine in this alternative work centre.

— Obviously, the loading situation and the idle start times of the machines in this alternative work centre will be different from the machine (which sets up the scheduling state) in this operation's original work centre. Hence, the tool availability constraint and the precedence constraint of this operation need to be checked again based on the new situation. As the alternative work centre may consist of several machines, this check will first be on the machine whose idle time starts the earliest and if those constraints cannot be satisfied on this machine, it will try the other machines.

— After a machine in the alternative machine centre has been found, the system will schedule this operation on this machine. The scheduling action and the reinforcement schedule updating are the same as the normal scheduling, which will be discussed later.

The most important characteristic of this substep is finding alternative machines to reduce the load in the heavily loaded periods. But that is only one aspect, taking earlier action (putting an action earlier than strictly required) is the other. To illustrate, let us take a very simple example to see how RESS-II takes early action to avoid any later problems. Let us say, the system is going to schedule on M3a and there are 6 operations waiting to be scheduled on this machine in the present state. The current partial schedules

and reinforcement plan analysis report that work centre 3 is currently in a bottleneck period. But no problem exists for any one of these operations to be scheduled on this machine at the moment, i.e. any one of the operations can be scheduled on this machine now without overloading the work centre in the current period. Of course, RESS-II can choose an operation to be scheduled on the machine according to the preference constraints, but remember we want to ease the load on this work centre. So it will decide to find alternative machines (in different work centres) at this early stage. The important reason for taking this early action is that later operations may not be able to find alternative work centres when they really need to because of some constraints. This ability to take early avoiding action due to informed looking ahead is an interesting capability. At the moment, RESS-II does not consider the alternative operations (or alternative routings), although it is not difficult to include this (choosing the routing that will not use the overloaded work centre will also ease the loading condition).

If the alternatives are found, before going to the next substep the system has to go back to substep (b) in step (1) to check the tool-availability constraint again on the operations left since those operations that have been scheduled on alternative machines may reserve the tools that some of these remaining operations need. This is a constraint check which makes sure the rest of the operations are valid for the following substeps.

(b) Classify the ready operations

The orders of the set of operations passed on from the last step can vary a great deal in their urgencies at the present state (set up in (1)). This substep uses the SLACK rule [24] to classify these operations into three categories: (1) very urgent group, (2) normal group, and (3) earlier group. Then each operation is weighted according to its category to produce an urgency rating of its order at this scheduling state. This rating is attached to the operation and passed to the next substep.

(c) Evaluate a given hypothesis on preference constraints

There are two preference constraints in RESS-II, the set-up time constraint and the machine preference constraint. At this stage, only the set-up constraint's satisfaction is considered. In this substep, the system first hypothetically puts each operation being considered onto the machine and then evaluates each hypothesis according to the satisfaction of this preference constraint, which is expressed by a pre-set weight. This weight is then incorporated into the existing rating of the operation.

However, the weight here does not play an important role in competing with the other constraints in the decision making. In fact, it is relatively unimportant and only has effect when other constraints' evaluation on the alternative choices produce almost the same rating, and when there is no resource problem at this scheduling state (although this process is for solving resource problems, these problems do not exist all the time and at all the work centres) so that the solving of these important problems will not be affected.

At first sight, in the constraint decomposition approach, this substep is seemingly unnecessary as the satisfaction of the local preference constraints depends on the local situations and the schedule refinement will consider them eventually. But this is only true of the machine preference constraint as its satisfaction does not alter a schedule much. The change of machine on which an operation is scheduled in the same work centre will be able to satisfy this constraint. There is no need to change the sequence of operations on machines. We say this constraint is at the machine level. Hence, it can be effectively satisfied in the schedule refinement (or the fourth stage) and does not need to be considered here. However, the satisfaction of the set-up preference constraint may need such big changes that the sequence of operations could be affected on a machine. This kind of change is very dramatic and very likely to influence the satisfaction of the other constraints in the schedule refinement. This is not allowed in order to prevent

interactions with the other constraints. So, the change in schedule refinement is limited only to the machine level. This substep helps to bring those operations with similar set-ups together at the machine level (but will not influence the solving of other problems). The same substep also exists in the main scheduling (the third stage).

(d) Deal with resource problems

The operations with their urgency and preference constraint ratings coming from the last substep is still not sufficient for a final decision. The problem remains of choosing a single operation from this group to be scheduled on the selected machine. This substep further evaluates each hypothesized schedule in order to solve some potential resource problems. It is a look ahead process. In fact, this and the taking of earlier action in finding alternative machines in the first substep characterize the decision making in the reinforcement approach.

Finding alternative machines to directly reduce the load on the bottleneck work centres is not sufficient to tackle the problem. The global bottleneck work centres can be underloaded in certain periods of time, and the local bottleneck periods may be leveled or balanced on its own work centre. So, the balancing of the load on the problem work centres is another important way to tackle the resource problem. In addition to the bottlenecks, other problems may also exist, such as congestion of less important resources in certain periods of time. To avoid or reduce the effects of all these problems the system needs to carefully sequence the operations on each machine. In RESS-II, these problems are expressed as dynamic constraints which are represented as look ahead rules aimed at solving these resource problems. These rules define the context of the resource based problem situations, and also the strategies open for dealing with them. The idea is that the system searches through the reinforcement level of schedule and the current partial schedule to determine whether those potential problems exist and can be solved if the operation

considered is scheduled next on the machine. If this is so, the attached weight associated with each rule is added to the rating from previous substeps. It can be expressed like this:

if <a problem situation exists>
and <it has not been solved>
and <this operation helps to tackle it>
then <add weight to existing rating>

After all the operations in the set are examined by these rules, the ratings are compared and the operation weighted most favorably is chosen to be scheduled next. For example, a rule may say: try to patch the possible future idle time periods on bottleneck work centres (both global and local) to balance the load on them. The problem situation is like this: an operation's next operation in the same order will be in a bottleneck period of a work centre while its previous period on such a work centre is underloaded, so if the current operation can be performed earlier, its next operation can also be scheduled earlier, hence the idle period may be occupied. Consequently, the load in that bottleneck period is eased. In applying this rule at a scheduling state, the system checks to see if this problem situation exists for the operation currently being analysed. If it does, the weight attached to this rule is added to the existing rating of this operation.

In applying a rule in order to detect and solve a resource problem, the system first needs to check whether this problem has been solved or not by previous decisions since a resource problem may be resolved in different ways and by different decisions. For example, the resource problem discussed in the last paragraph can be solved in at least two ways. One is the obvious way, i.e. because of the earlier finish of the previous operations of some operations in the bottlenck periods, these operations in bottleneck periods can be performed earlier or shifted to an earlier time period so that the load in the bottleneck periods are balanced. Another is the one discussed in the last paragraph. Since there are many operations in a bottleneck period that can be shifted to

their previous periods, earlier decisons involving shifting some operations backward may balance the load. If this is the case, the current decision making does not need to consider it again. This is done by checking the finish time of the last operations of the operations in the bottleneck period.

(3) Resource reservation

The schedule decision made in the last step and passed to step 3 is the final form. A particular operation has been selected to be performed next on the selected machine. The resource time bounds have been associated with each selected resource. This step establishes the reservation for each required resource in the schedule. The resulting resource reservations will act as additional constraints for the subsequent scheduling. An example is shown in Figure 6.7, which has machine M2a and tool T1b reserved for operation Op2a2. According to the schedule representation, the operation is put on the machine and the tool is reserved. So, there are two processes; one is putting the operation on something and another is reserving tools. The intentions are different.

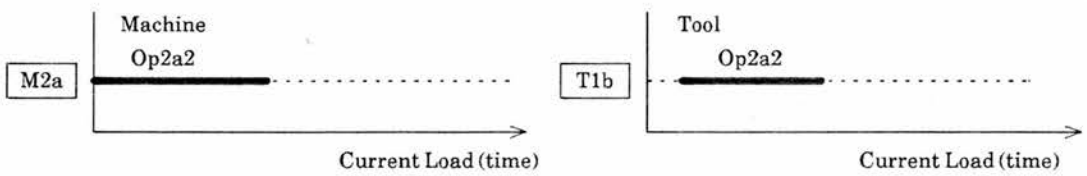


Figure 6.7 A sample resource reservation

(4) Update the reinforcement level.

This final step maintains the validity of the reinforcement schedule which the detailed scheduling consults. It is essential for reliable inter-stage analysis and communication. The updating action changes the reinforcement level in terms of what has been scheduled and what has been left at the present time so that the reinforcement schedule also reflects the current detailed scheduling decisions. For example, when an operation is scheduled before its time period (specified by its lead time bounds) in the detailed scheduling, the system has to

remove the amount of load from that period and add to the current period in the reinforcement schedule. Such an updating step and the subsequent reinforcement level analysis in the following scheduling cycles gives the system the ability to dynamically detect high-contention, normal loading and underloaded areas in the shop. Consequently, it provides a clearer picture for the decision making. (This decision of updating of the reinforcement level will not affect earlier decisions made.)

6.2 The Main Scheduling Stage:

— Schedule Construction

After those two stages of reinforcement building have finished, the system has sufficient information to do the schedule construction. The conflict situations between the due date constraint and the resource availability have been shown in the capacity plan and the conflict situations between the due date constraint and the work-in-process constraint has been shown in reinforcement two (from the second stage of reinforcement building). The objective of this stage is to take these two schedules as reinforcements to generate a schedule which will reduce the work-in-process time and also maintain the due date constraint's satisfaction from the second stage. The scheduling method here is also the parallel method.

Although pushing forward or backward and moving operations around were discussed in chapter 4, they are not actually done in the original schedules (which is now the reinforcements) in RESS-II. Instead, they are performed in a new partial schedule created for the new process. The reason for this is that the schedules of operations are inter-related in many senses. Changes made to the schedule of one operation will typically have some influence on the schedules of the others. It could be extremely difficult for a system to track such a process.

This scheduling process is quite similar to the one at the previous stage: setting up scheduling states; making decisions by current scheduling situation analysis and reinforcement schedule analysis; updating the reinforcement. But, the differences are that the extra work-in-process constraint is considered at this stage, and that for this purpose one more reinforcement is used, the scheduling result from the second reinforcement building stage. As discussed before, the conflict situations or problems between the work-in-process constraint and the due date constraint are very difficult to predict at the previous stage, but after that stage, the situation becomes much clearer. By accessing this schedule, the system will be able to recognise such problems. Joining force with the other resource allocation rules used at the second stage, this scheduling process can make better choices, which not only shortens the work-in-process time but also maintains the due date constraint's satisfaction result.

Although the capacity plan is also used as the reinforcement here, this stage will not try to find alternative machines for operations. The alternative machines found at the previous stage for some operations were recorded, i.e. their original work centres have been replaced by the alternative ones. In this scheduling, these operations will still be scheduled on the machines in those alternative work centre.

The following gives a list of the steps in this process. The ones that are the same as at the previous stage will not be further discussed.

(1) Find the ready machine and the operations

This step first finds a ready machine to set up a scheduling state, then finds the ready operations that are waiting at this machine.

(2) Scheduling decision making

This step evaluates the operations from the previous step and chooses a single operation to be scheduled.

(a) Check the work-in-process time

This substep finds out the orders (the operations belong to) that wait for too long in the reinforcement two (from stage two) and assesses how long each of them can be pushed forward to a later time. Only the reinforcement two is used here.

Although the system evaluates each individual operation, in the process, it considers the order of each operation as a whole. Since the work-in-process time of an order is only reflected by the start time of its first operation and the finish time of its last operation, in reducing the work-in-process time, the effect will only be on the first and the last operation of the order rather than the rest of them. In RESS-II the base point for reducing the work-in-process time is set at the finish time of an order from the reinforcement two since this point is more or less fixed. As mentioned before it is the long waiting operations that influence the work-in-process time; they cannot be pushed backward as this may violate some order's due date. After these operations, the schedules will seem normal. So, only the first operation will be considered here, i.e. try to push forward the start time of each order to reduce the work-in-process time.

This substep first gets an operation from the operation set passed over from the previous step and checks if it is the first operation. If it is, the system will extract all the rest of the operations in the same order from the reinforcement two and find out if there are any operations that have gone through bottlenecks, causing their waiting in the queue to be too long. If such operations are found, the system will see how far each of the operations' previous operations in the same order can be moved forward. This forward move in time is added to the existing start time of the operation, which becomes the new hypothetical start time of the order. If the operation is not the first operation, it will not be considered. After all the current operations

are analysed in this way, a result is produced which tells what operations can be pushed forward and how long they can be pushed forward.

(b) Preference constraint evaluation

This substep assesses the satisfaction of the set-up time constraint on each operation.

(c) Evaluate all the operations

This substep is the same as "deal with resource problems" at the previous stage, which includes assessing the nature of the operations, such as urgency and importance, and evaluate them according to those resource problem rules by using the capacity plan as the reinforcement. The only difference is that at this stage, this substep records the operations whose schedules can solve some resource problems, such as by balancing the load.

(d) Consider the work-in-process time

After all the operations have been evaluated in the previous substeps, the system is ready to put the work-in-process time constraint into force to further assess the possible schedules of these operations. Unlike the consideration of the other constraints, reducing work-in-process time is handled here in a very different way. This substep will not rate each operation according to the satisfaction of this constraint. Instead, it eliminates the operations that can be scheduled later from the set of candidate operations passed on so that they will not be considered for a final decision. This is essential to this scheduling process. In order to do this, the system uses the result from substep (a) and the recordings from substep (c) above. After that, the system makes the final decision to choose a single operation to be scheduled.

To eliminate those operations that can be scheduled later requires an assurance that such an elimination will not cause any resource problems, otherwise, the due date constraint's satisfaction from the previous stage may be violated. For this purpose, the information (stating which operation's schedule will solve the potential problems) recorded in the previous substep will be used. The system checks each operation in the resulting list of operations (that can be pushed forward) from the substep (a) of this step against the recordings to see whether it will influence resource allocation. If the operation being considered will not harm the resource allocation, it will be eliminated from the original operation set. If it will, it will not be removed. After all the operations in the list have been checked in this way, a subset of candidate operations is produced.

From this subset the system makes the final decision to select an operation to be scheduled. The operation which is rated most favorably will be chosen. In the case that every operation can be eliminated, the most urgent one will be chosen to be scheduled.

(3) Resource reservation

This step reserves the resources that the selected operation needs.

(4) Update the capacity plan

As at the previous stage, this step updates the capacity plan to make it also reflect the current scheduling situation. The reinforcement schedule two will not be updated as this stage maintains that schedule but only pushes the orders with long waiting times forward to reduce the work-in-process time.

6.3 Schedule Refinement

This stage refines the schedule constructed by trying to satisfy as many local preference constraints as possible, which will in turn improve the schedule. The fundamental idea is to shift the operations around on the same machine or on different machines in the same work centre to satisfy the local preference constraints without violating global ones.

(1) Finding the ready machine and the operations

This step first finds a ready machine to set up a scheduling state, then finds the ready operations that are waiting at this machine.

(2) Scheduling decision making

(a) Select a number of operations to be considered

Not all the operations that have been passed on need to be considered in this local optimisation. Instead, only a number of urgent operations should be evaluated. This substep selects these operations according to the number of machines in the work centre and the reinforcement schedule. Those which are not urgent at the moment will join the later evaluations.

(b) Preference constraint satisfaction analysis

The selected operations from the last step are analysed one by one to see how the preference constraints are satisfied. In such an analysis, the system first hypothetically puts each operation being considered onto the machine and then evaluates each hypothesis according to the preference constraints. As a constraint is typically relevant to a particular class of scheduling decision, the first thing in constraint evaluation of a given hypothesis must be to identify the set of constraints that should participate in the process. This is

done by collecting the constraints from the operation description and the global constraint representation. The evaluation is performed by rating the constraints according to their satisfaction. Each constraint has a pre-set weight attached to it. The rating for each hypothesis is attached to the operation. After all the operations have been analysed the results are passed on to the next step for further evaluation.

(c) Choose an operation to be scheduled

In this step, the system finds out which operation can be best scheduled here so that more local preference constraints can be satisfied. This is done by firstly sorting all the operations in a descending order according to their preference constraint satisfaction rating and then hypothetically putting the best operation on the chosen machine and the others on the rest of the machines in the same work centre. (The operations' start times from the result of the reinforcement are also sorted, as are the current end times of the machines in the work centre. Each operation is put on the corresponding machine.) The system then accesses the reinforcement schedule to see whether the start time of any of the operation's next operation in the same order will be violated in this hypothetical situation. If there is no problem, then the current operation will be chosen to be scheduled here. If there is, then the second best will be tried in the same way and so on. In the case where none seems correct, the most urgent one will be selected.

(3) Resource reservation

This step reserves the resources required by the selected operation.

6.4 Summary

This chapter described the whole schedule construction process. The basic idea is to predict and deal with the resource problems and the constraint conflict problems. It consists of four stages of scheduling. The two reinforcement building stages build two reinforcement schedules. By analysing these schedules, the main scheduling stage is able to obtain a global insight to foresee problems and deal with them accordingly to generate a schedule. At the refinement stage, this schedule is further refined in order to satisfy more local preference constraints. The next chapter will discuss monitoring and rescheduling which is the function of the fifth stage of RESS-II.

Chapter 7

Shop Floor Monitoring and Rescheduling

After a detailed schedule has been constructed, it will be used on the shop floor to control the production process. However, this schedule is simply a forecast. In real processes many unexpected situations may occur, such as machine breakdown and operator absence, etc, which may invalidate and force changes to be made to the existing schedule. This chapter focuses on this issue, i.e. monitoring and rescheduling, which is the function of the fifth (the last) scheduling stage of RESS-II. Given the current situation and the existing schedule, this stage makes "real-time" decisions and updates the existing schedule. During the process, it also attempts to retain as much of the existing schedule as possible.

In discussing the problems, unexpected situations in the job shop environment are identified, which highlights the demand for a new scheduling process, rescheduling, to deal with them. But apart from these interruptions, as a scheduling process, rescheduling introduces its own problems; it needs to maintain shop stability and to consider the constraints that were originally handled in the schedule construction processes. It is shown that what is needed in decision making at this stage is an analysis

of the current situation and the overall situation, and a consultation of the existing schedule to combine them to make decisions. That is what the reinforcement approach provides, so reinforcement scheduling also forms the central technique here. However, the constraint decomposition approach will not be used. As a result, the implementation of this reinforcement approach is different from that in the schedule construction. In fact, two reinforcements are used in the rescheduling decision making: the capacity plan from the first stage and the final (or existing) schedule.

7.1 Unexpected Interruptions on the Shop Floor

The schedule constructed from the last stage is in its final form and can be used to control the production processes on the shop floor: operations' time bounds (start time and finish time) within which they will be performed have been set up; machines and tools they require have also been reserved. According to this schedule, the shop activities will be planned and the whole production process will be carried out. However, in real world situations, things may not proceed precisely as planned or scheduled. Many unexpected problems may occur during actual manufacturing, machines may break down, operations may be started and finished later or earlier than scheduled.

The following is a list of possible shop floor interruptions.

(1) Earlier or later start and finish of an operation

Although an operation has fixed time bounds in which it should be started and finished, deviations are inevitable for various reasons. Normally such changes of start or finish times are so small that they will not influence much of the existing schedule. The schedule does not need to be changed as a result. But in some situations, because of operator absenteeism or an unskilled worker being assigned to perform the task, such deviations can be so significant that the system has to change the existing schedule.

(2) Machine breakdown

Machine breakdown is a normal problem. No one can expect a machine to run forever without any interruption putting it out of work. Such a situation will make it impossible to perform the operations that have been scheduled on the broken down machine in this time period. Thus, they have to be rearranged according to the new situation.

(3) Tool resource non-availability

Besides machine breakdown, tool non-availability also causes disruption. Like the machines, tools are also reserved during schedule construction. However, when an operation is due to be processed according to the existing schedule problems in tool preparation or problems with other operations may cause the tool reservation and the existing schedule to become invalid.

(4) Operator absence

An operator may be absent during the manufacturing process due to various reasons, such as sickness and lateness, etc. This prevents execution of the operations that should be performed during the time period. It is very similar to machine breakdown interruption.

(5) Quantity change

Every product has its required quantity. In scheduling, this influences the process time of each operation, but due to market change or customer requests the management may decide to reduce or increase the quantities of certain products. This means that the loading times of the affected operations in the existing schedule will be changed and the loading conditions on the machines will be altered.

(6) New orders

Job shop production is a continuous process; new orders continue to arrive and finished products are continually produced. Moreover, some orders may be cancelled by the management. Therefore, the cancelled orders have to be removed from the existing schedule and the new orders have to be added.

(7) Due and start time changes

Because of management decisions or problems with material handling, the due and start times of some orders may be changed. This may also influence the existing schedule.

These problems are unpredictable. They cannot be foreseen in schedule construction (which only attempts to schedule the jobs with respect to the production plan and existing resource situation at that time). The schedule constructed (the existing schedule) is only a forecast of what is likely to happen on the shop floor. It cannot take into account any unplanned interruptions. However during the real manufacturing, these problems do occur, and they may invalidate and force changes to be made to the existing schedule. To keep up with the progress of the manufacturing process, monitoring the production and rescheduling or updating the existing schedule becomes very important. To deal with such a situation, a new scheduling function is required, namely, monitoring and rescheduling, further complicating the real world scheduling task. In fact, the capability of any scheduling system is also measured by how well it can respond to changes, that is, how efficiently it can reschedule and reload the work in response to what is actually happening at any given time.

7.2 The Conflict Situations Caused

In the above, the possible interruptions on a shop floor have been presented, but they do not explain what conflict situations may be created in terms of scheduling. In the

following, we identify these conflict situations and classify them into four groups. This classification forms the framework for the system to deal with them.

(1) Precedence constraint violation

This conflict corresponds to the situation where the precedence constraint of an operation has been violated such that the operation cannot be performed as it was scheduled. The cause for this is that, due to various reasons, an operation's previous operation has finished so late that its finish time has exceeded the start time of this operation. Every interruption can lead to this type of violation.

(2) Resource capacity conflicts

These conflicts correspond to the situations where the resources (which include machines and tools) are not available when they should be. Machine breakdown and tool non-availability are two obvious examples. Operator absence also causes this problem. In addition, they also account for the situations where the capacities required are not available, such as when adding a new order to the existing schedule.

(3) Time duration change

These conflicts come from the situation where the duration of an operation has been changed. They will result in idle or extra time in the existing schedule. For example, quantity change, earlier or later finish of operations and cancellation of orders could produce this kind of problems.

(4) Schedule objective change

These problems correspond to the situation where the criteria on which the existing schedule is based is altered so that the existing schedule will not be valid. Such problems may include due date and start time changes.

7.3 Problems at the Rescheduling Stage

The last two sections identified job shop interruptions and the conflict situations they may create. This section will go into more detail to analyze these problems and their impacts on the existing schedule. Along with this, other problems in rescheduling will also be addressed.

We assume that the schedule constructed represents the best possible compromise among conflicting constraints. Indications of changes in the shop floor status provide information as to the actual nature of these constraints. Hence, the goal of rescheduling will be to find the problems according to the shop status and to update the existing schedule. As with any change, this could lead to many situations: situations where changed shop status introduce inconsistencies into the existing schedule and situations where the changed status offer opportunities to improve the existing schedule because it may remove some of the constraints. In the first case, the schedule has become infeasible, and reactive revision must centre on restoring feasibility. In the latter case, the feasiblity is not a problem but rather it is advantageous to revise decisions to better satisfy factory constraints.

We will first discuss the situations where changed shop status introduce inconsistencies into the existing schedule. Let us take "machine breakdown" as an example to illustrate such interruption situations and their effects on an existing schedule (the other problems have similar effects). It will have two obvious effects on the existing schedule. The first one is that the operations that have been scheduled on it in the same period cannot proceed. The second is that it may split the operation that was performed on it when it was down as the operation may not be finished at that time. These effects invalidate the existing schedule. The system will need to reconsider the schedules on the machine, pushing some operations forward until the machine has been repaired and shifting the other operations to other machines in the same work centre. But as the operations in a schedule are typically connected with each other by different constraints, such changes (pushing and shifting) will not normally be limited to one machine and not even to one work centre. In other words, they may also influence the

schedules of the rest of the operations in the same work centre and in the other work centres. As a result, the rest of the schedule will also need to be updated accordingly. Below, two figures illustrate such a situation. Figure 7.1 shows the existing schedule before machine M1b has broken down. Figure 7.2 shows the possible rescheduling results after the machine is down. The down machine M1b is shown in Figure 7.2 as a thick black line and the length of this line represents the possible down time. Operation Op1a1 is split and the remaining part is rescheduled on machine M1a. The dashed line means that the machine is idle during that period and the non-dashed line means that the machine is loaded.

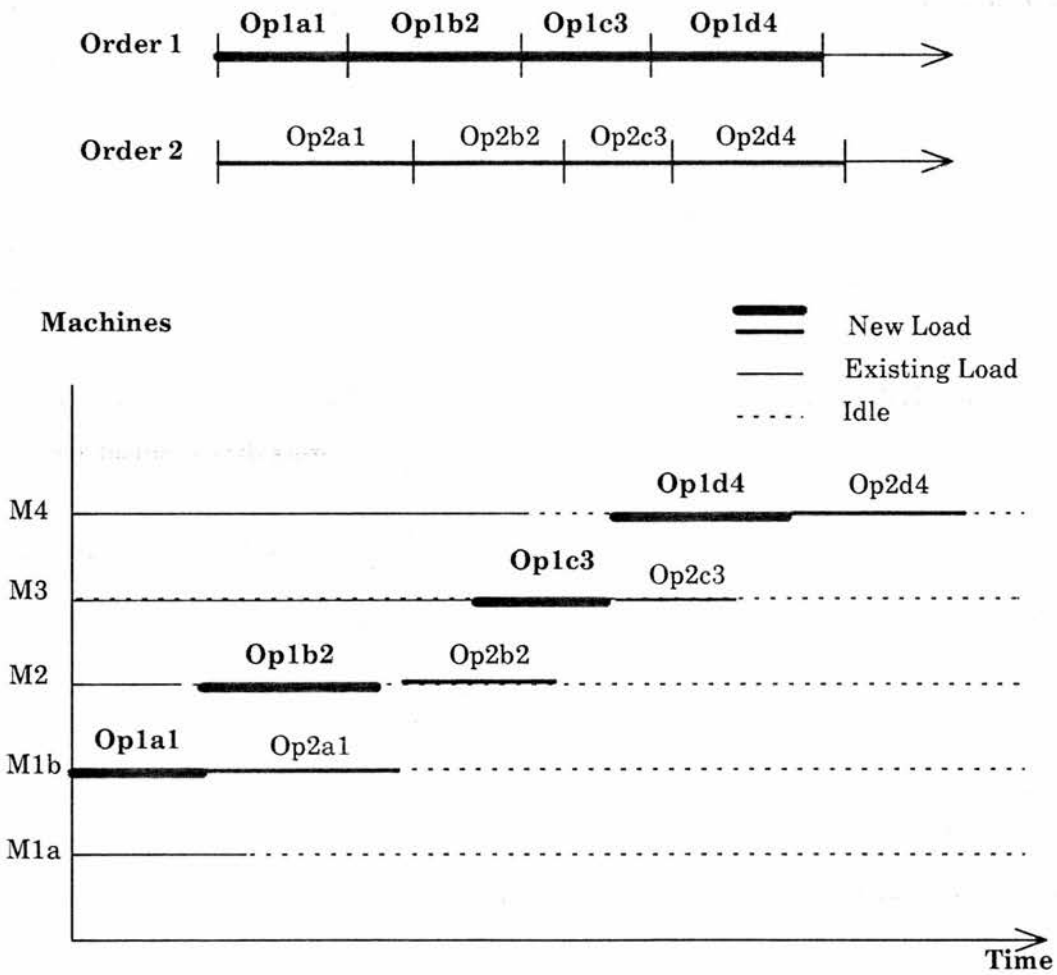


Figure 7.1 The original schedules

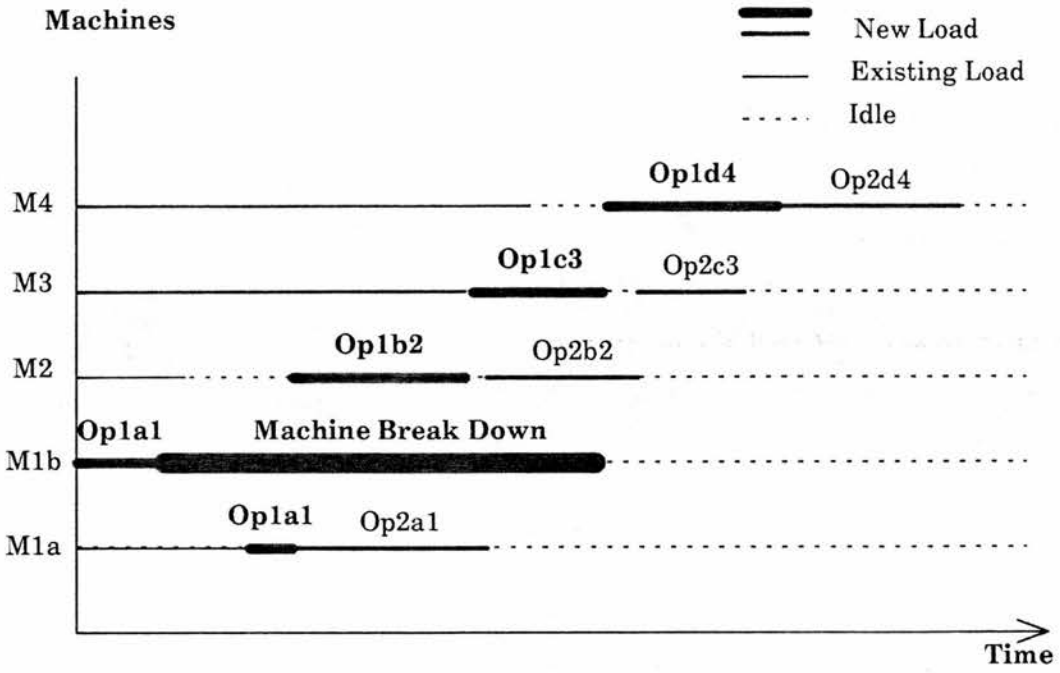


Figure 7.2 The possible schedules after M1b has broken down

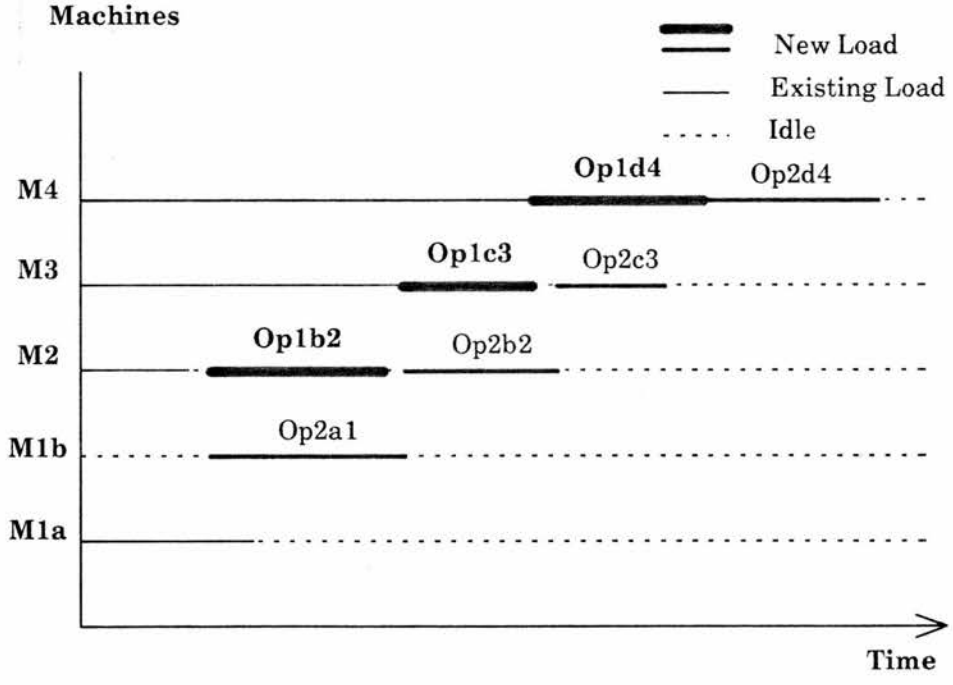


Figure 7.3 The possible schedules after the load on M3 is finished earlier

Unexpected situations can also have a positive effect on the existing schedule. For example, the load on a machine may finish earlier than scheduled. Figure 7.3 shows such a situation. The operations before Op1c3 on M3 have finished some time earlier than they were scheduled (see the original schedule in Figure 7.1). As a result, operation Op1c3 can be performed earlier and so can Op1d4, Op2c3 and Op2d4. Consequently, both orders are finished earlier.

It is true that rescheduling deals mainly with unexpected situations, but as a process it carries other problems along with it that must be handled.

One such problem is to maintain shop stability, that is to make as little change to the existing schedule as possible. A schedule does not just tell when the operations should be performed on their machines, more importantly it guides the preparation tasks and other shop activities. A changed schedule will result in a changed shop activity plan. However, a working shop is a physical environment. Any changes to it will take time and need effort. Hence, it may not be able to respond to too many and too frequent changes in a schedule, which may cause problems. For instance, last-minute changes may lead to increased periods of machine idle time as the preparation cannot be done quickly enough.

Another problem is the control of how much scheduling the system should perform after each interruption point. In actual manufacturing, unexpected situations may happen at any time. If a shop has many machines the interruption will become even more acute. It will not be practical to expect a system to schedule all the operations after every interruption. Such a process will not only take too much time and too much computational effort but is also unneccessary. In fact, the shop situation may change so fast that the system cannot keep up with the pace of such changes if full scheduling is performed. Instead, after the interruption point, only a part of the scheduling needs to be accomplished to guide production in the near future.

Apart from the new problems discussed above that need to be dealt with in the monitoring and rescheduling process, all the old problems and constraints which have

been considered in the scheduling construction processes will still exist at this stage. Let us stick with the machine breakdown example described earlier. Since work centre 1 has two machines, immediately after machine M1b has broken down, the system will reconsider the next choice on machine M1a from a set of operations waiting. This waiting list is obviously not the same as the waiting list from the schedule construction process since a part of operation Op1a1 and operation Op2a1 are added. At this point, if we compare this decision making with the one in schedule construction, we will find that the problems of these two are almost the same: those constraints considered in the schedule construction will still be considered here, both objective constraints and preference constraints; global perspective will also be the essential requirement. The differences are that a special operation, i.e. machine breakdown, has been loaded on machine M1b to occupy a period of time and that an extra constraint on shop stability has to be considered.

7.4 Dealing With Rescheduling Problems

In this section, the problems discussed in the last section which need to be tackled in the rescheduling process are analysed to see how they can be dealt with.

The first problem, still the main problem, is maintaining a global perspective. It is the same as in schedule construction, so the reinforcement approach will also form the central technique at this stage. As the rescheduling process and the second stage scheduling construction process are very similar (except that more problems are considered in rescheduling), the capacity plan from the first stage (the reinforcement building stage) is again used here as a reinforcement. The difference is that interruptions are represented and reflected in this reinforcement plan as special loads or new loads. One need not think that the rescheduling techniques will be characteristically different from those used to generate a schedule.

The second problem is to maintain the shop stability, that is to change the existing schedule as little as possible. To deal with this problem, the system will use the existing

schedule as another reinforcement and turn the resource reservation in this reinforcement schedule into a preference constraint to participate in the rescheduling decision making process. Its satisfaction is also weighted as with the other constraints.

The third problem is that only a part of scheduling should be performed after each interruption, rather than all of it. It is actually a simple control problem and can be easily done by setting a time limit which the rescheduling process should respect. How long this period should be depends on different shop situations and it is easily changeable.

Although the reinforcement approach is still the central technique in this rescheduling stage (process), the constraint decomposition approach will not be used. So, unlike the schedule construction stages, all the constraints at this stage are considered in this single rescheduling process. There are two reasons for this. Firstly, the time required to do so. A constraint decomposition approach satisfies different constraints in the different scheduling processes. In the schedule construction, three single scheduling processes are performed and they are done completely. This takes a long time. It will not be suitable for "right now" decision making at this stage. In fact, there may be so many problems on an actual shop floor that the system will not be able to cope if three complete scheduling processes have to be performed whenever a problem occurs. Secondly, there is no need to use the decomposition approach as far as the basic schedule layout and work-in-process constraint are concerned since the shop stability constraint at this stage will keep as much of the existing schedule layout as possible.

Because the schedules or the positions of operations in a schedule are connected by time, there is not very much difficulty in detecting the problems, but to update them in the existing schedule could be a big problem. As we discussed before, it is very confusing and not easy for the system to track. In RESS-II, on finding the first problem, the system makes the existing schedule the reinforcement and schedules again to generate a new schedule to replace the old one.

7.5 Handling the Unexpected Problems

Before going on to discuss the monitoring and rescheduling function in RESS-II, the handling of these unexpected problems needs to be addressed. They reflect the way they are expressed in a schedule.

The unexpected situations can be divided into two groups. The first group of problems do not need to be further represented in the existing schedule. Earlier or later start and finish of certain operations only express the deviations to their schedules. Updating of the schedules affected will be sufficient to represent such situations. New orders, quantity changes and due or start time changes only cause the original data to be altered according to the new values. However, the second group of problems, such as machine breakdown and tool non-availability, need to be represented as they cannot be reflected in the existing schedule representation. In RESS-II, machine breakdown is represented in the existing schedule as a special forced operation which means this operation has to be performed now on this particular machine for a certain period of time and it requires no tool. As for tool non-availability, there may be two reasons for its cause. The first is that some operation that uses the same tool as the current one requires, is still in process when it should be finished, i.e. it has been delayed, and there is no extra tool available. So, the current operation cannot be performed. The second reason is that the tool has broken down or is not ready when it should be because of the delayed preparation. In the first situation, updating of the schedule of the delayed operation will reflect the tool non-availability for the current operation. But in the second situation, a representation is required before the system can change the tool reservation, and it is represented in the same way as machine breakdown.

7.6 Monitoring the Manufacturing Process

In order to make the adjustment needed to ensure the closest possible conformance with the schedule constructed, the system must have a steady flow of information about the status of orders and the reasons for any deviations from the schedule. In other words, the

progress of operations and orders must be monitored so that delays and shop problems can be spotted and remedied as quickly as possible by changes of schedule and production assignments so that the shop is kept running with a minimum of disturbance and lost time. This is the first function of this final stage of scheduling (the second function is rescheduling). The following information is required for this objective.

(1) Where the operation takes place	Work centre Machine number
(2) Operation identification	Order number Operation number
(3) Quantity produced	Quantity completed
(4) Time consumed	Start time Finish time (current time)
(5) Problem encountered	Breakdown Operator absence Resource non-availability Quantity change Due and start time change New order

(As discussed in the last section, earlier and later start or finish of an operation do not need to be represented in the problem category. Reporting of the information in (1) to (4) will reflect any deviation to its original schedule).

There are two kinds of reporting in RESS-II; normal reporting and problem reporting. Normal reporting reports the progress of the operations: which operation has been loaded on which machine and when it started and when it finishes. This reporting will cover the problems of earlier and later start or finish of operations. Problem reporting specifies the problems in category (5) above to the system. In RESS-II, the reporting scheme uses a frame-based representation.

Normal reporting is represented by a frame:

[Operation-state:
Order-number:
Operation-number:
Work-centre:
Machine-number:
Operation-start-time:
Operation-end-time:
]

Note: Operation-state slot describes whether the operation has finished or just started. Order-number and Operation-number specify the particular operation. Work-centre and Machine-number tell where the operation is performed.

Unlike normal reporting, the problem reporting scheme has different slots for different problems. For example, machine breakdown is reported to the system as this:

[Breakdown
Work-centre:
Machine-number:
Order-number:
Operation-number:
Process-time-left:
Present-time:
Expected-back-time:
]

Note: Work-centre and Machine-number specify the down machine. Order-number and Operation-number specify the operation that is currently being processed on the down machine. Process-time-left shows how much process time is left for this operation. Expected-back-time tells when this down machine is expected to be back in service.

7.7 Rescheduling in RESS-II

Rescheduling in RESS-II is also a reinforcement scheduling process. It is very similar to the original process of constructing a schedule, but because of its special needs, two reinforcements are employed — the capacity plan from the first stage and the existing schedule. All the constraints are considered in this single scheduling process. The situation in such a rescheduling is like this: given the current situation in the shop, the system first updates the capacity plan (reinforcement one) and the existing schedule (reinforcement two); then it decides whether rescheduling is needed; if it is, rescheduling is performed until a certain time limit is reached. Figure 7.4 illustrates this process.

(1) Inputing the current situation.

During the manufacturing process, some person or computer monitoring the process can specify what has been performed and what problems have happened up to the present time to the system. This is the monitoring process.

(2) Interpreting problems

The user need not report all things that have changed since many of them may be deduced by the system's operators. Many effects can be deduced from certain critical problems, for instance, if a machine has broken down, there is no need to report that the operation currently being processed on this machine has been split if it has not yet been finished.

(3) Changing the existing schedule to reflect the current situation

According to the current situation input in the first step, the system changes the existing schedule so that it also reflects the effects of the current situation. For example, consider the case where a new operation has just finished later than expected and a machine has broken down. In the first situation, the operation's

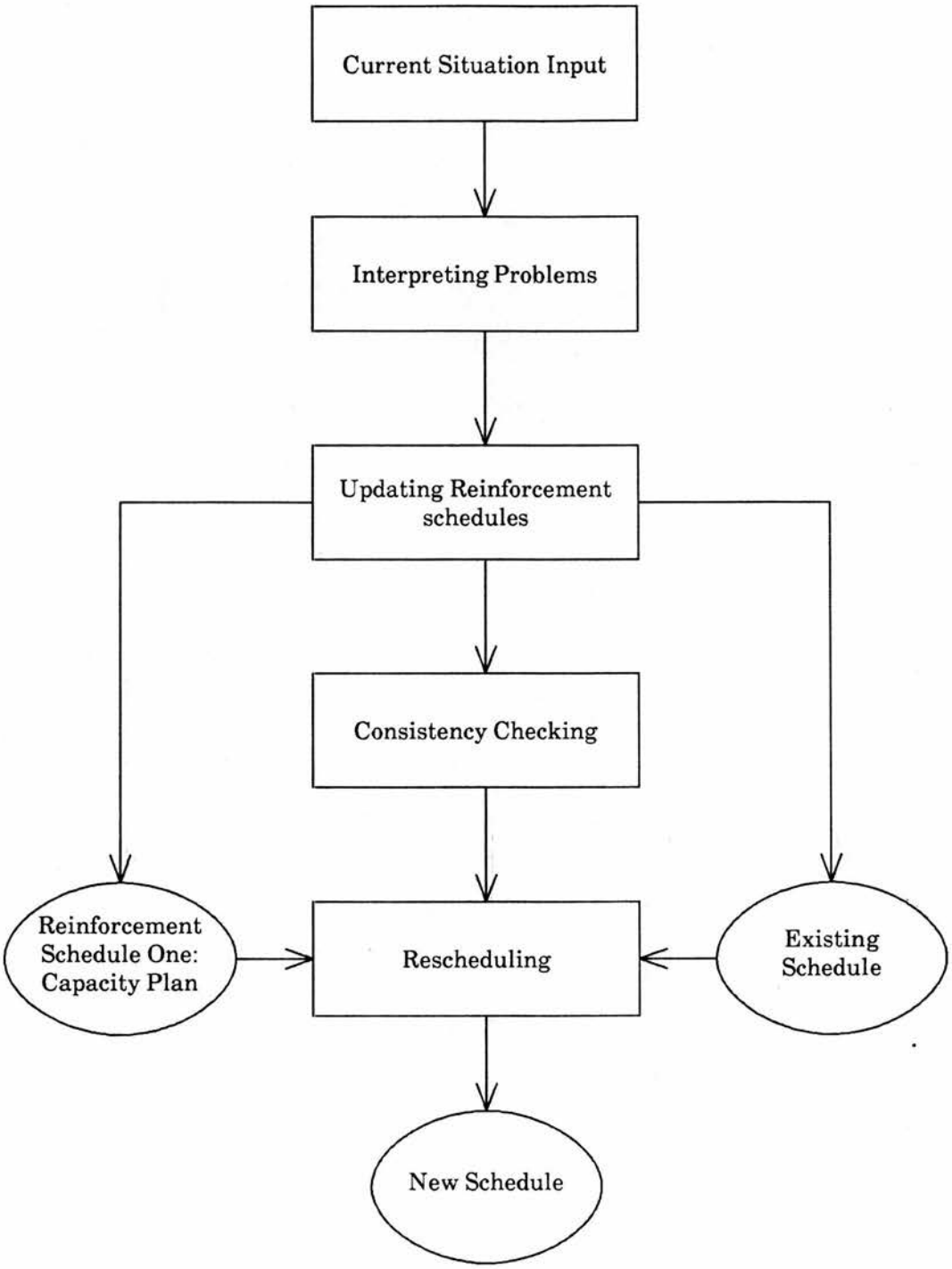


Figure 7.4 Rescheduling architecture

position in the existing schedule will be updated according to the new finish time. In the second situation, the breakdown will be represented and put in the existing schedule. This step does not change the rest of the operations in the existing schedule.

(4) Updating the reinforcement schedule one

As the rescheduling process will use the first reinforcement building stage result (capacity plan) to guide the decision making, this step will update this result according to the current situation. For instance, if a machine has broken down, the system will add the breakdown time to the load of that work centre in the period (as the infinite scheduling does at the first reinforcement stage). This is very similar to the process of updating at the second stage of reinforcement building.

(5) Checking whether rescheduling is needed.

After the current situation is input and the updatings have been done, the system checks if there are any problems with the existing schedule and whether rescheduling is needed. This is done by accessing the existing schedule to see if the remaining schedules can still be used in the new situation. If some of them cannot, rescheduling will be performed. Otherwise, the system will do nothing but wait for the next input. In fact, the checking will stop and a signal raised to the effect that rescheduling should be performed if any single operation cannot be performed as scheduled. It does not go on to find all the problems.

(6) Rescheduling process

Since the interruption has caused the existing schedule to become invalid, rescheduling has to be performed. It will be done in a newly created current partial schedule (which has a copy of the current situation from the existing schedule) rather than the existing schedule (as discussed in chapter 4, changing

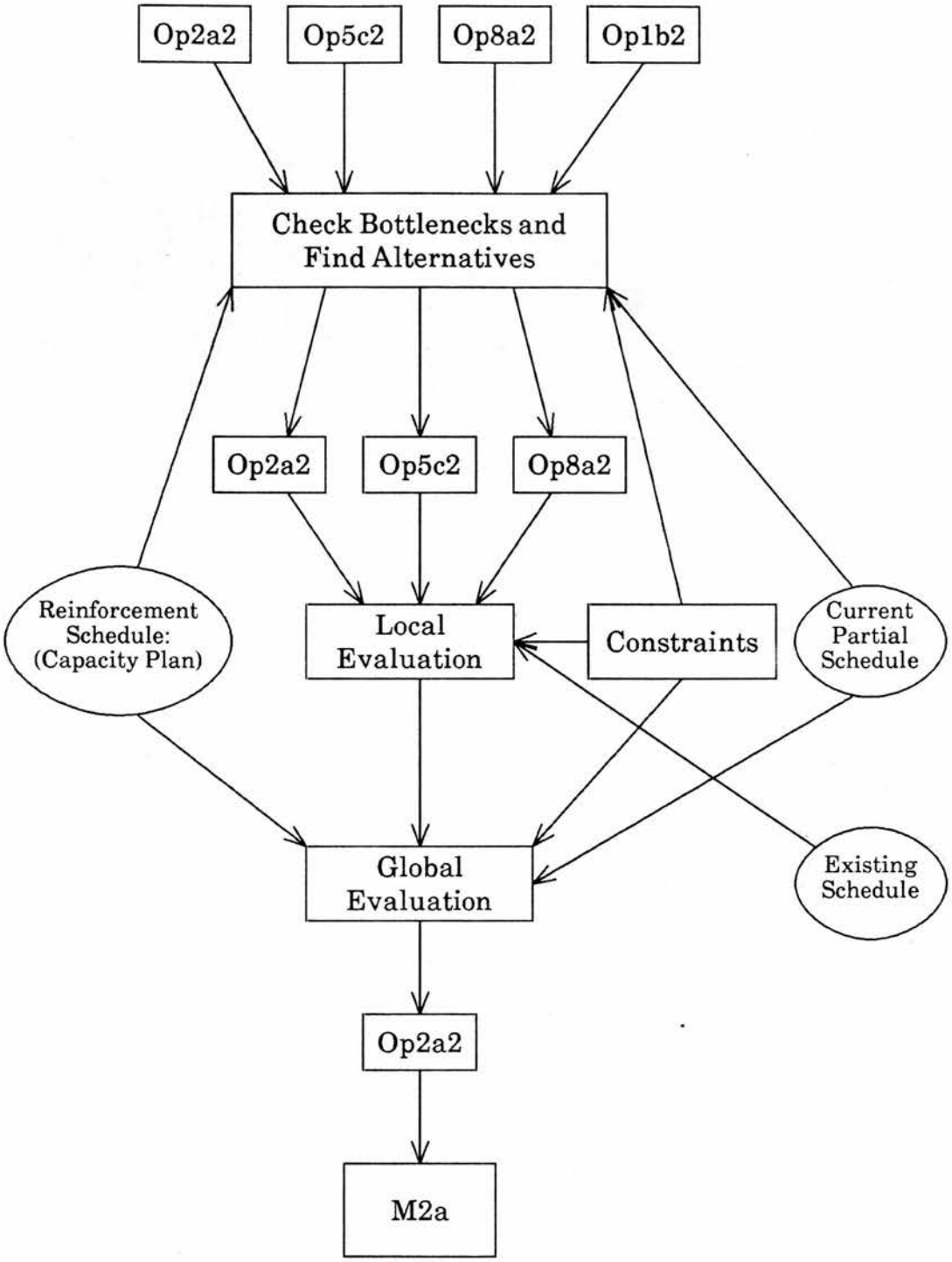


Figure 7.5 Reschedule decision making

an established schedule is very difficult). After the rescheduling has been finished, this current partial schedule becomes the existing schedule and the old existing schedule will be destroyed. Figure 7.5 shows the rescheduling process.

There are two rescheduling strategies in RESS-II, intended for different situations. Strategy one is for scheduling on the machines in the work centres directly affected by unexpected situations and strategy two is for scheduling on the machines in the rest of the work centres.

Before starting the rescheduling of operations, the system first decides and initially marks the work centre(s) that will be directly affected by the unexpected situations. This is a deduction process. The information needed would have to be provided as part of the specific description. The criterion for making such a decision is whether the schedules (of the operations) in a work centre will definitely be changed. For example, if the unexpected problem is that a machine has broken down, then the work centre that this breakdown machine belongs to will be marked as a directly affected work centre and the others will be the indirectly affected work centres. Some of the interruption situations do not signal any directly affected work centre at the beginning, but they may do so dynamically during the scheduling process. For instance, if a new order has arrived and it needs to be scheduled now, i.e. for processing right away, it does not signal any directly affected work centres. However, in the scheduling process, when any of its operations appears in the waiting list at a work centre, this work centre will be marked as a directly affected work centre then.

The decision making in scheduling on a machine in a directly affected work centre has two actions and uses two reinforcements (strategy one). The first action is similar to the one used at the second stage of reinforcement building, i.e. analysing the current situation and reinforcement one to produce a rating. The difference is that all the constraints are considered in this single scheduling process. The second one involves looking up the existing schedule to find the operation in the same position. If this original operation is found, it will be

attached with a special weight to express the consideration of the shop stability constraint. The final decision is made by combining these two ratings.

The decision making in scheduling on a machine in an indirectly affected work centre only has one action and uses one reinforcement, the existing schedule. In such a situation, the system finds out the original operation in the existing schedule in the same position and tries to re-schedule it without any extra analysis. In the case that this operation cannot be re-scheduled, the system will have to apply strategy one and this work centre will be dynamically signalled as a directly affected work centre (this decision only affects the later scheduling).

Treating the scheduling in different work centre in different ways not only helps to maintains shop stability but also reduces the computation effort of the complicated analysis.

7.8 An Example

This section presents an example to show how the monitoring and rescheduling stage of RESS-II works. Again we take machine breakdown as an unexpected problem.

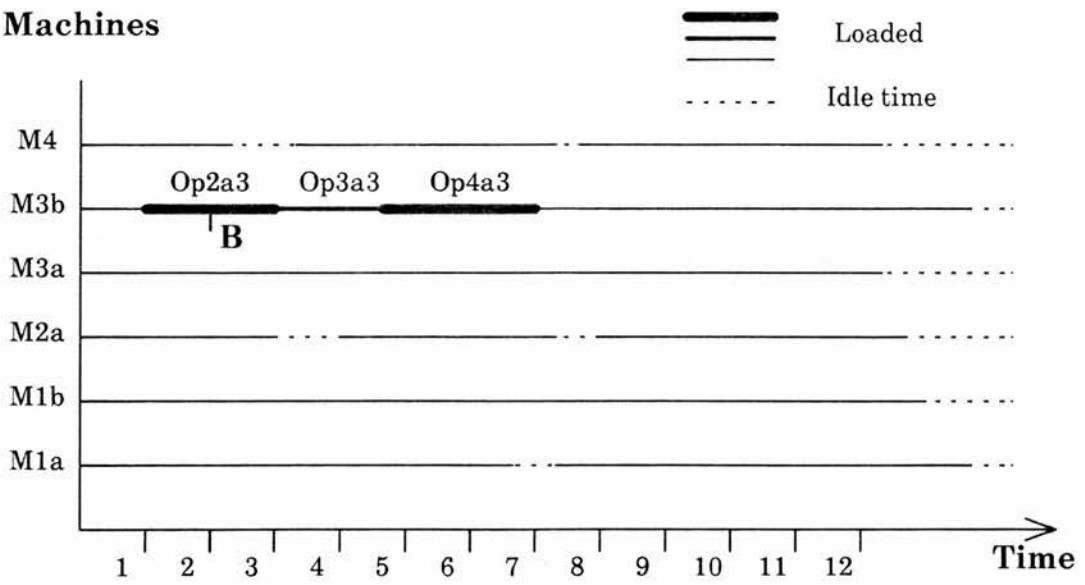


Figure 7.6 The existing schedule

Figure 7.6 above shows the existing schedule (only three operations are specified and the rest of them are not). Now, suppose machine M3b has just broken down at time "B" (in Figure 7.6).

(1) Inputing the current situation

As the problem is machine breakdown, to specify this to the system we will use the interruption representation discussed in section 7.6. It can be input to the system like this:

```
[Breakdown
    Work-centre:            3
    Machine-number:         b
    Order-number:           2
    Operation-number:       a
    Process-time-left:      1
    Present-time:           2
    Expected-back-time:     6
]
```

(2) Interpreting problems

Two problems will be interpreted from this interruption. The first one is that the breakdown machine will be out of work for a period of time, i.e. be occupied by a special operation (described below). The second one is that the schedule of operation Op2a3 has been split into two parts: a complete part (that has been processed) and a remaining part. The remaining part will be put in the waiting list at this work centre (work centre 3).

(3) Updating the existing schedule.

This breakdown is represented in the existing schedule as a special operation like this.

[Order-number: breakdown
Operation-number: breakdown
Start-time: 2
End-time: 6
]

This operation is also put in the current partial schedule on the breakdown machine in the time period from 2 to 6. The graphic representation is shown in Figure 7.7.

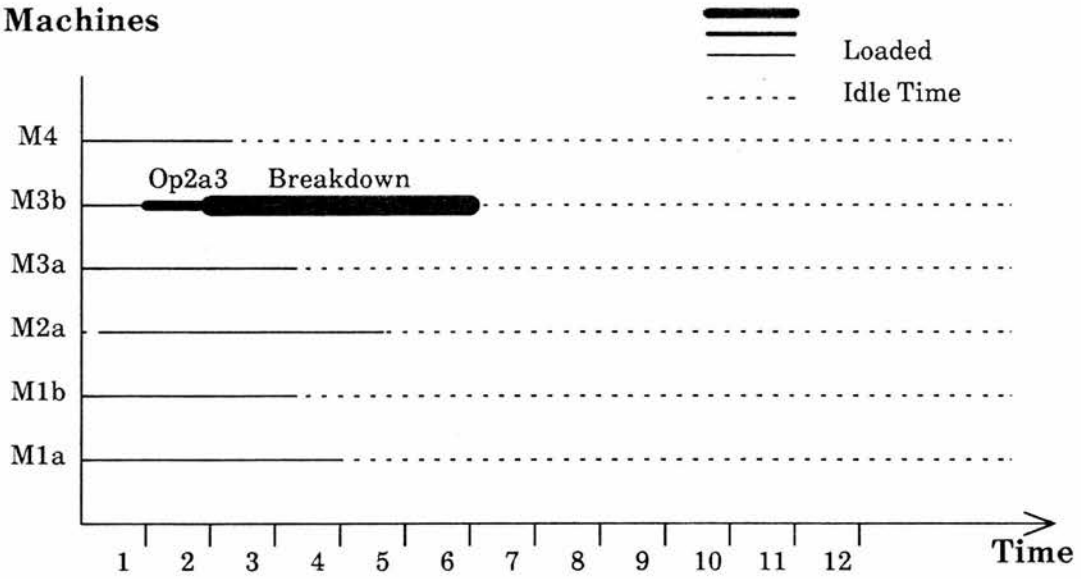


Figure 7.7 The updated existing schedules so far

(4) Updating the reinforcement one.

First, the capacity plan (from the first reinforcement building stage) is retrieved. Then, the breakdown time period is loaded onto the work centre that the breakdown machine belongs to. The loading method is the same as the updating of the reinforcement plan in stage two of the reinforcement building. Here, two time periods are affected by this breakdown: the first and the second time periods. Figure 7.8 shows the process and updated capacity plan.

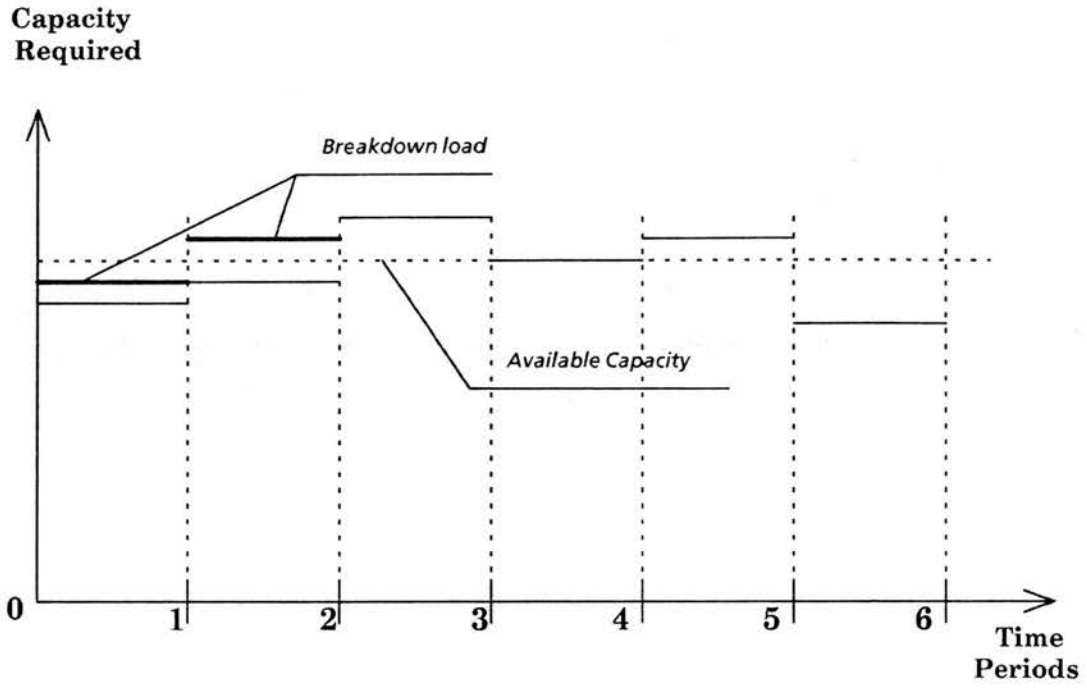


Figure 7.8 Updated capacity plan:
the reinforcement schedule one

(5) Checking whether rescheduling is needed

The problem will be that operation Op2a3 cannot be performed as it was originally scheduled. Two other obvious operations that will be affected directly are Op3a3 and Op4a3. But as soon as RESS-II detects the problem with Op2a3, it will not try to find any more problems.

(6) Rescheduling process

Work centre 3 is signaled as a directly affected work centre at the beginning. After rescheduling has been finished, a possible schedule could be like the one shown in Figure 7.9. In this schedule, the unfinished part (or remaining part) of operation Op2a3 is scheduled on machine M3a in the same work centre (work centre 3). Operation Op4a3 is scheduled after the breakdown and Op3a3 is scheduled onto the other machine. The rest of the schedule is also changed but is

not shown in this diagram. Finally, this new schedule replaces the old existing schedule to act as the new existing schedule.

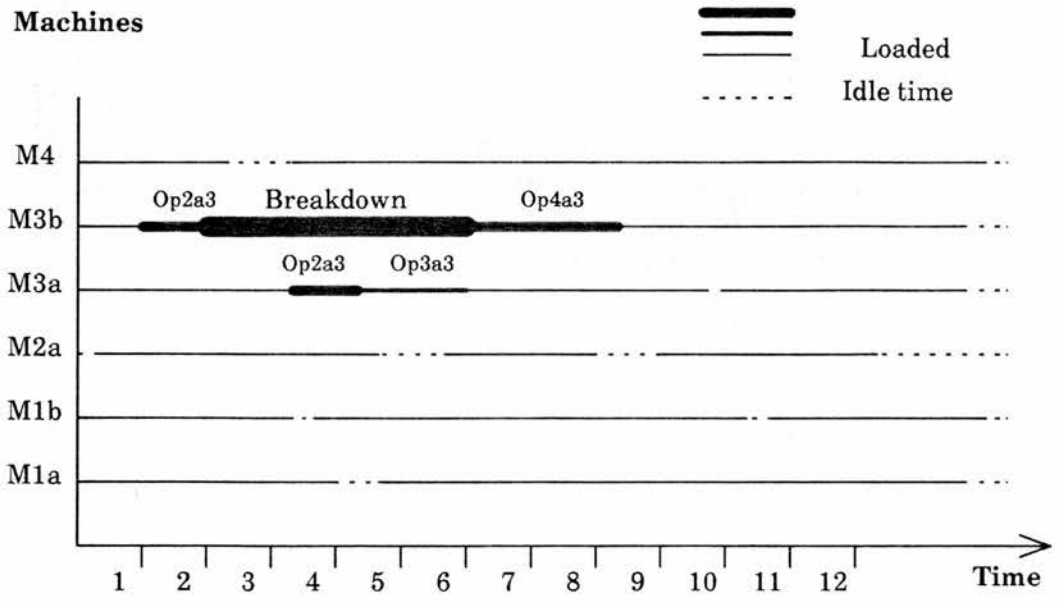


Figure 7.9 A updated existing schedule

7.9 Summary

This chapter discussed the rescheduling function of the RESS-II scheduling system. It is characterised by dealing with the dynamic nature of the job shop and performing "right now" decision making as a reaction to the unexpected situations in the schedule execution on the shop floor. This function differs from schedule construction in that it needs to consider the extra unexpected situations, while also keeping as much of the existing schedule as possible. This makes it more complicated. However, the technique used is still reinforcement scheduling, although the constraint decomposition approach is not employed.

This completes the description of the RESS-II system. From the next chapter we will present some experimental results and make a comparison bewteen this research and the work that has been done in the past.

Chapter 8

Experimental Results

In order to test out the techniques that have been presented in this thesis, a number of experiments have been conducted with the RESS-II scheduling system, which embeds these approaches. These experiments were performed in three shop environments. Within each environment 12 different tests have been carried out, and each test schedules a set of orders. Altogether, 36 sets of orders have been scheduled in three environments. This chapter will present these experiments and explains the schedules produced.

In the process, the results from different stages of RESS-II are compared to see how reinforcement scheduling and constraint decomposition influence the scheduling results and how effective they are in performing their tasks. Along with testing of RESS-II, we also employed the initial version of the RESS system, RESS-I [44] (recall that RESS-I only incorporates one reinforcement stage, and no constraint decomposition approach), in our experiments to further illustrate the advantage of constraint decomposition.

Since we have no access to other systems, such as ISIS [31] [32], in order to perform comparison, the results stated here offer a measure of performance of the RESS-II system.

8.1 Test Environments

As there is no actual factory case that is available to be used in this research, all the data is generated (by a separate program). However, the generation was based on a general factory model [24] [43]. Our experimental job shops work 8 hours a day and five days a week. Each shop consists of a set of machines, a set of tools and a set of process routings which describe how many kinds of product can be produced in the shop and what route each product should follow to be manufactured.

Different shop environments have a different number of work centres, and each work centre has a different number of machines, as do tools (all the tools, jigs, fixtures, etc. are grouped into one category under tools without being distinguished). To date, we have used three shop environments to test our system. The following lists the number of machines in these three shop environments.

Environment 1		Environment 2		Environment 3	
W/C	Machines	W/C	Machines	W/C	Machines
1	3	1	4	1	3
2	3	2	3	2	3
3	3	3	4	3	3
4	4	4	3	4	3
5	2	5	3	5	3
6	3	6	4	6	3
7	4	7	4	7	3
8	3	8	5	8	3
9	4			9	3
10	3			10	4
				11	3
				12	2

Note: W/C: work centre number
Machines: the number of machines in the work centre.

Environment 1 consists of 10 work centres. Environment 2 consists of 8 work centres and environment 3 consists of 12 work centres. Within each environment, there are a number of rough processing centres and a number of finishing processing work centres, which are in the earlier part and the later part of the machine list respectively. (There are also different numbers of tools within different environments, but as they are treated in the same manner as the machines they are not listed here.)

According to these three physical shop environments, we should be able to generate the routings for the products that can be produced in the shops. The first step is to decide how many routings should be generated. Since scheduling does not depend much on how many kinds of product a factory can produce but is heavily influenced by how many orders the factory has received, so in every experimental environment, the number of products (parts) that can be manufactured is the same, 200. The second step is to generate each individual routing according to the existing machines and tools in a particular environment.

In our experiments, the products that can be produced in the shops do not have any fixed product line or family, i.e. different routings do not have any relationship with one another. There is no alternative routing for each product either, i.e. only one routing is available for each part, but alternative machines for an operation can be used in the process. Each routing consists of a number of operations with a part number attached to denote this particular part. The number of operations in each process routing is in the range of 3 to 7. The sequence of operations in each order describe the precedence constraints (ordering constraints).

Within each operation, the necessary work centre, tool and other processing data are specified. In choosing the work centre for an operation, the basic rule is that earlier operations use earlier work centres (which are used for rough processing) and the later operations use later work centres (which are for finishing). The operations in the middle use the work centres in the middle. However, this is not fixed, and there are also exceptions. For instance, in some cases, a part may not need precision manufacturing and only rough processing will be sufficient. This is handled by assigning a small

probability against such a situation occuring. The selection of a particular work centre within a group is completely random as long as two successive operations do not use the same work centre. An operation may also be able to use an alternative work centre, different from its original work centre, and the alternative work centre is also generated in the data.

Generation of tools for operations is slightly different from generation of work centres. Firstly, different work centres may have different tools that are specifically used by them and some tools can be used on machines in different work centres. So, in choosing a tool for an operation, a special file containing the information about which tools are for which work centres is accessed and one is selected at random. Secondly, not every operation has to use a tool as it has to use a machine. So, only some of the operations will be assigned a tool. To achieve this, a special generator is used to decide which operation should use a tool and which one should not.

Running time and set-up time are also generated randomly. The running time for each operation is between 10 to 40 mins and the set-up time is between 30 to 120 mins.

The preferred machine for an operation is generated by selecting a particular machine within the work centre. As with tools, only a limited number of operations will need preferred machines.

Here is a sample process routing used in our testing, routing for part 14.

Part-No: 14

Oper-No	W/C	Alter	Tool	Setup/T	Run/T	M/pref
1	5	none	11	120	16	pref 2
2	2	none	3	60	12	
3	7	8	15	75	25	
4	10	none	19	60	15	
5	9	10	19	90	13	pref 1
6	10	none	0	75	16	

Note:		
	Oper-No:	operation number
	W/C:	work centre that the operation requires
	Alter:	alternative work centre
	Tool:	tool required
	Setup/T:	set-up time required
	Run/T:	manufacturing time required
	M/pref:	preferred machine in the work centre (if any)

Within each environment, the constraints considered are the ones that have been discussed in the previous chapters. They are:

(a) Meeting delivery due dates
(b) Reducing work-in-process time
(c) Increasing resource utilization
(d) Set-up
(e) Machine preference
(f) Precedence constraint
(g) Machine availability
(h) Tool availability
(i) Job importance
(j) Urgency

Since the testing here is not concerned with dynamic issues of the job shop, so the machines are expected to run without any interruption. No over-time working and extra shifts are considered either.

This describes how our experimental shop environments have been set up. We then need to generate orders to initiate requests for scheduling.

8.2 Generation of Orders

Orders are requests for certain products. They can come from different sources, such as from customers and forecasting. There are also different kinds of orders, which influence due dates and importance values of the orders. In this testing, three kinds of order are considered:

- Forced-outage orders

  These kind of orders have very short lead time and need to be manufactured as soon as possible. They normally have higher importance values. In our testing, they account for 15% of the total orders.

- Stock orders

  These kind of orders are to be placed in the stock for future needs, so their due dates are not so important. They account for 10% of the total orders.

- Normal orders

  These kind of orders have longer lead times for the factory to process. Their importance varies according to different situations.

However, the orders used for our experiments are not the ones that have just been received from customers or forecasting, but have been processed by management. Hence, they are actually direct data for scheduling and contain extra information about management decisions, such as start times and importance values of orders. Altogether, an order for the test contains the information about the part required, the quantity required, start time, due time and order importance. The generation of orders is performed by manipulating these values. Each of these values has a special range within which they can be selected and the selection is completely random. Different orders can ask for the same part .with the same quantity or with different quantities. The number of orders generated for each test is between 120 and 150. The following further explains each value and presents how it is selected.

- Part number

  This specifies which particular product an order requires. It may be any one within those two hundred parts that can be manufactured in our factory. The selection is random.

- Quantity

  Orders are released in batches. The batch size of each order is also generated randomly. This and the running time of each particular operation decide the load time of this operation. Consequently, it plays an important part in deciding the loading condition of the machines. In this test, it ranges from 10 to 50.

- Start date

  This refers to order release date. Our test release is based on a week, i.e. the orders are released at the beginning of each week. The start time intervals between different orders are also selected at random from week 1 to week 7 (where the current week is week 1).

- Due date

  As the release time has been chosen, the due date can also be interpreted as order lead time, i.e. the time between order release time and due time. There are two things that need to be considered in deciding this. One is what kind of order it is and the other is how many operations the part has (or what is the minimal machine time as the lead time cannot be shorter than this time). For the forced outage orders, the lead time should be very short and for the normal orders and the stock order, the lead time should be longer.

- Importance

  This is a management assigned value, ranging from 1 to 10. 10 represents the most important orders and 1 represents the least important orders. It is decided according to type of order. For instance, the forced outage orders normally have high importance values and the stock orders have low importance values. Within each group, the importance values are generated randomly.

An example order could be like this:

Order-number:	23
Part-number:	14
Start-time:	week-5
Due-date:	week-9
Importance-value:	10
Start-operation:	3

Note: Start-operation denotes that, at the beginning of scheduling, the operations before this specific operation (in this case it is 3) have already been performed and only the remaining operations in the orders need to be scheduled. This is used for creating a continuous scheduling environment and will be discussed in the next section.

8.3 Test Results

8.3.1 Continuous Testing

There are two kinds of testing that can be performed in a shop. One is static testing and the other is continuous testing. Static testing assumes that before a set of orders are scheduled the shop is empty, i.e. no orders are being processed. ISIS [28] [31] adopts this testing paradigm, but this is unrealistic because in any real situation, new orders keep coming in and finished orders are being sent out. There are always some orders in process. Continuous testing will take into account such a dynamic situation.

To simulate the effect of a continuous shop condition, extra orders are generated during the order generation. These extra orders have some of their operations performed before the current schedule date. They are also included in the total number of orders and account for 20% of the total orders. These special orders are scheduled in exactly the same way as the new orders.

8.3.2 The Results

This sub-section presents all the test results from different experiments in three shop environments. Within each scheduling environment, twelve tests have been conducted (each test schedules a set of orders). The results are reported for each test rather than aggregating them across tests. These results will also be analysed according to the satisfaction of constraints, during which two versions of the RESS system and different stages of RESS-II will also be compared. (recall the difference between the RESS-I and the RESS-II systems is that RESS-I did not implement the constraint decomposition approach so that all the constraints are considered in a single detailed scheduling process.)

Let us consider first the tardiness of different results. Since the tests within the first and the second environments do not produce many overdue orders (actually, most of the tests have no late orders), we will not go any further to discuss them. However this is not to say there is no bottleneck work centre in these two sets of tests. In fact, there were some overloaded work centres in every test. The reason why there is no overdue order produced is that these overloaded work centres were used by the operations in the middle of a routing, and later operations were able to catch up, or that the bottlenecks were balanced in the scheduling process.

However, the tests in the third environment clearly have overdue orders. The load in a bottleneck work centre varies between 85% of its capacity and 125% of its capacity and in each test there may be more than one global bottleneck work centre and a number of local bottleneck time periods.

Figure 8.1 shows the performance of the two scheduling systems on average tardiness. We can see that the average tardiness does not change much with different systems and even with different stages of RESS-II. The reason for this is that although RESS-II has more stages of scheduling than RESS-I, only its second stage (which is almost the same as the detailed scheduling in RESS-I) tries to satisfy the due date constraints and the other stages only try to maintain this satisfaction result. However, we can notice that

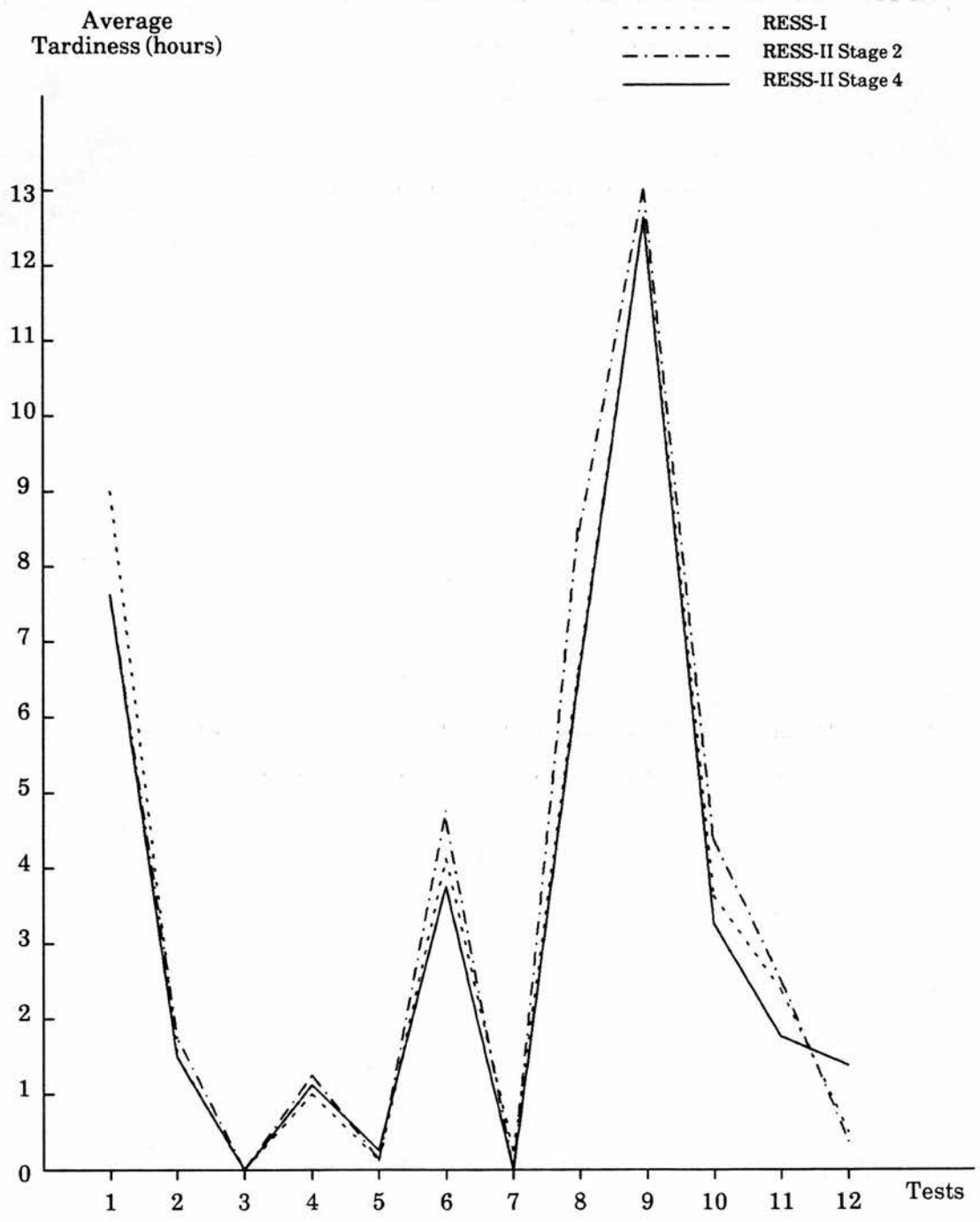


Figure 8.1 Average tardiness per order over different tests (Environment three)

the fourth stage of RESS-II shows a slight overall improvement over the second stage of RESS-II. This is because more set-ups have been saved by stage four, which improved the loading condition.

The following table shows the actual number of overdue orders over different tests (also in environment three). From this we can also see that there are 23 more orders that were finished on time in the results from RESS-II than from RESS-I, and 19 more orders that were finished on time in the results generated by stage 4 of RESS-II than stage 2 of RESS-II. Four exceptions were also produced, which have more overdue orders in stage 4 than in stage 2 of RESS-II (they are test 2, 5, 9, 12). The reason for this is that scheduling is such a context dependent problem that it is almost impossible to have an identical result after a new scheduling has been performed. However, such deviations are quite small (except test 12), and in most cases our approach is effective. So, it is reasonable to say that the satisfaction situation of the due date constraint is maintained after those different scheduling stages.

Tests	Number of Overdue Jobs		
	RESS-I	RESS-II-2	RESS-II-4
1	43	38	35
2	22	21	23
3	0	0	0
4	19	18	16
5	2	2	3
6	25	27	25
7	2	1	0
8	31	35	30
9	40	40	41
10	40	42	33
11	34	32	26
12	12	7	12

Note: RESS-I: the results from the RESS-I system

RESS-II-2: the results from the second stage of RESS-II

RESS-II-4: the results from the fourth stage of RESS-II which is also the last stage in schedule construction.

Figure 8.2 shows the average work-in-process time per order over different tests in environment one, Figure 8.3 shows the situation in environment two and Figure 8.4 shows the situation in environment three. These three figures compare the results from RESS-I and RESS-II. We can see that the work-in-process time has been dramatically reduced by RESS-II, but the result from the second stage of RESS-II is nearly the same as the result from RESS-I with respect to the work-in-process time. The explanation is that although the work-in-process time was considered in the decision making in RESS-I (though not taken into account at the second stage of RESS-II), because the future situations cannot be predicted, decision making has to concentrate on the satisfaction of the most important constraint, i.e. the due date.

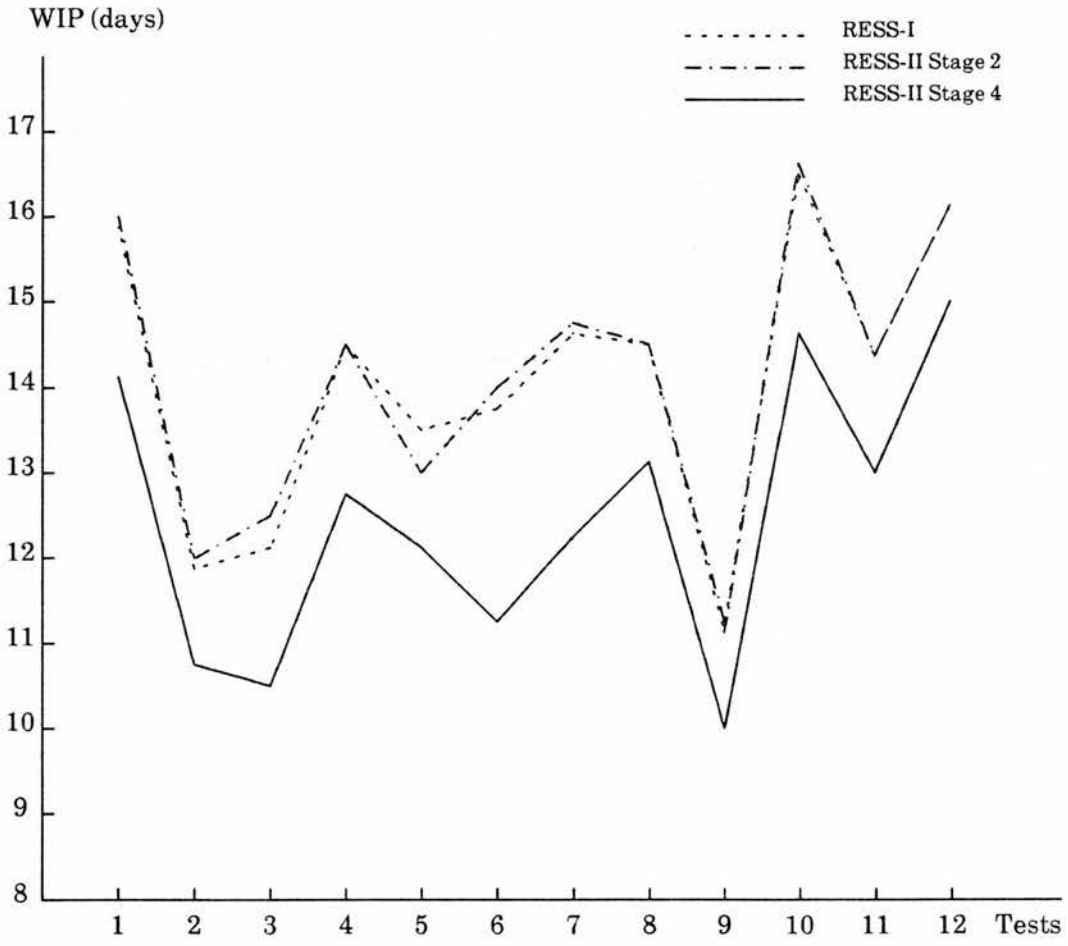


Figure 8.2 Average work-in-process time (WIP) per order over different tests (Environment one)

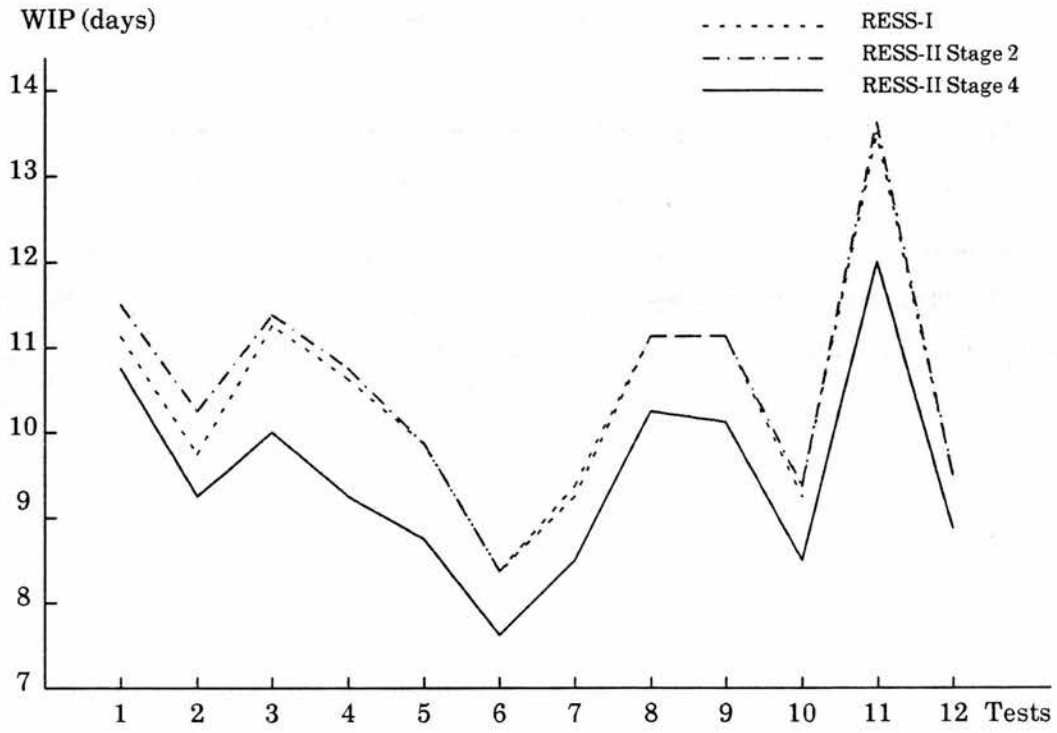


Figure 8.3 Average work-in-process time (WIP) per order over different tests (Environment two)

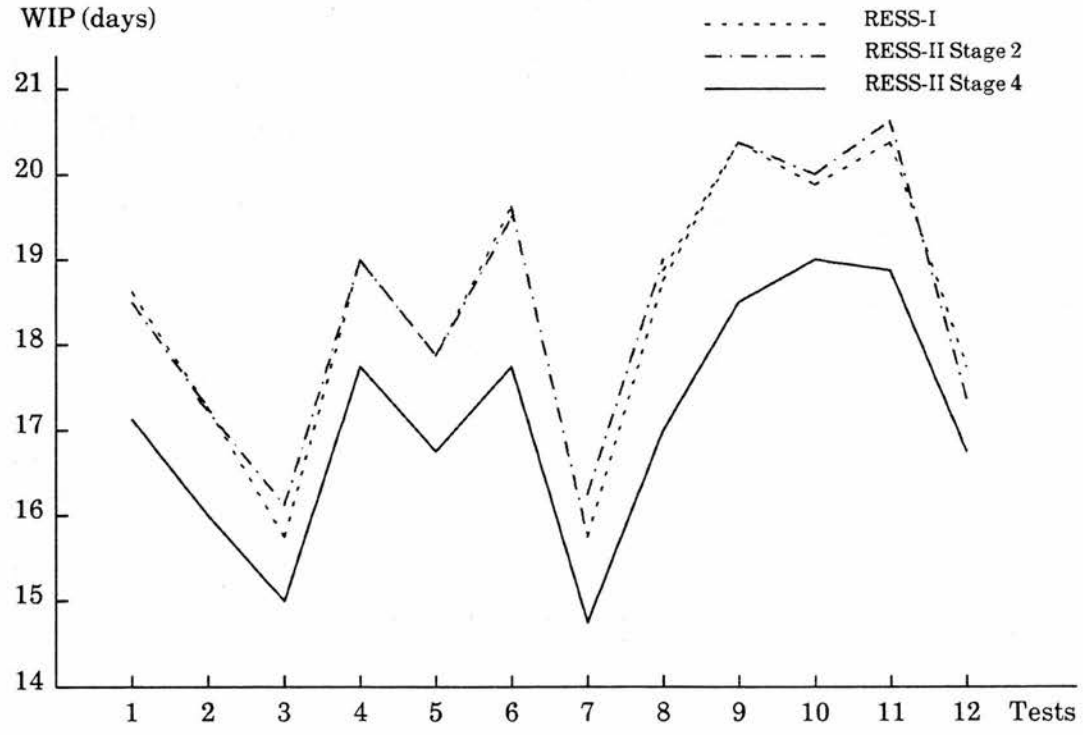


Figure 8.4 Average work-in-process time (WIP) per order over different tests (Environment three)

The number of local preference constraints that have been satisfied is also substantially increased by RESS-II. Figures 8.5, 8.6 and 8.7 show the situations. We can see from these that the result from stage 4 of RESS-II, which is also the final stage of the schedule construction, has greatly increased the number of set-ups saved. It can also be noticed that stage 4 produced an improvement over stage 3, which shows the effectiveness of the refinement stage in RESS-II. However in Figure 8.7, an exception is also revealed. The reason is also due to the problem of context dependent. It is very difficult to find a way that will produce the best result. So, we cannot say that the techniques suggested in this research can guarantee the best result, i.e. there could be some other better results which may not be generated from these techniques. However, in most of the situations, these techniques are able to produce good results.

The next three figures, Figures 8.8, 8.9 and 8.10, show how many machine preferences are met with RESS-I, stage 3 and stage 4 of RESS-II. Again, RESS-II continues to show a marked improvement with the inclusion of constraint decomposition and later refinement to the schedule.

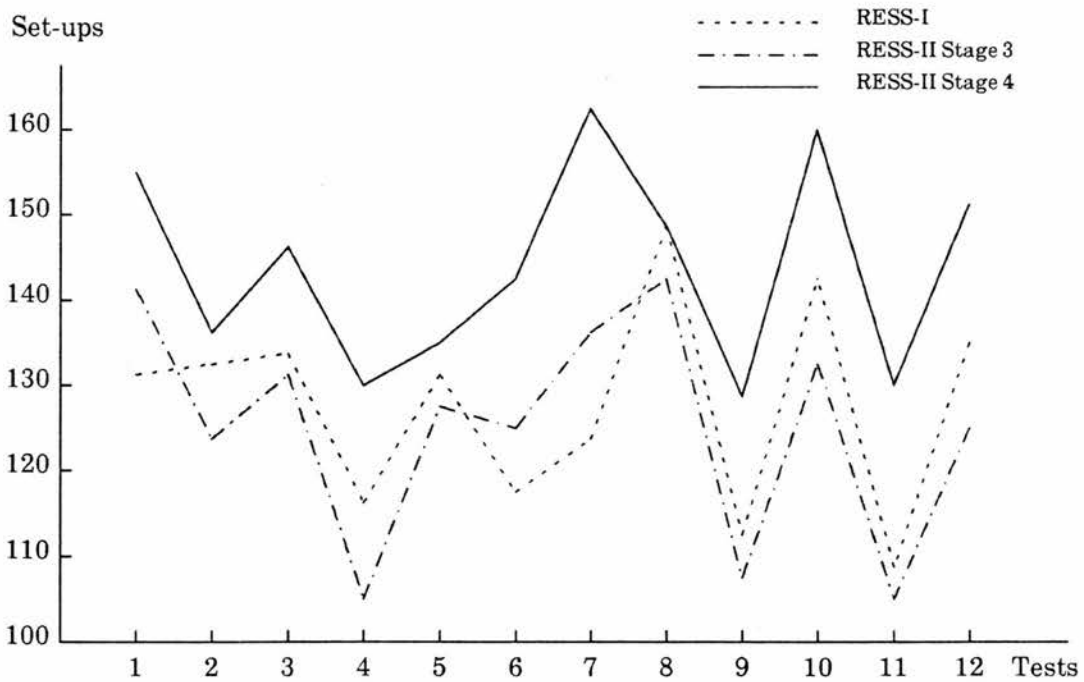


Figure 8.5 Number of set-ups saved over different tests (Environment one)

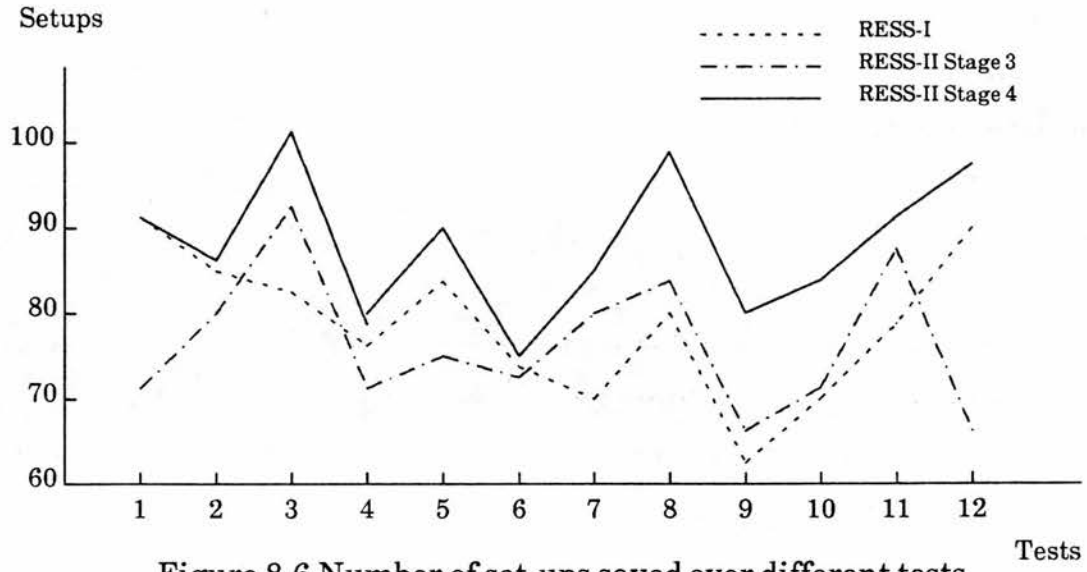


Figure 8.6 Number of set-ups saved over different tests (Environment two)

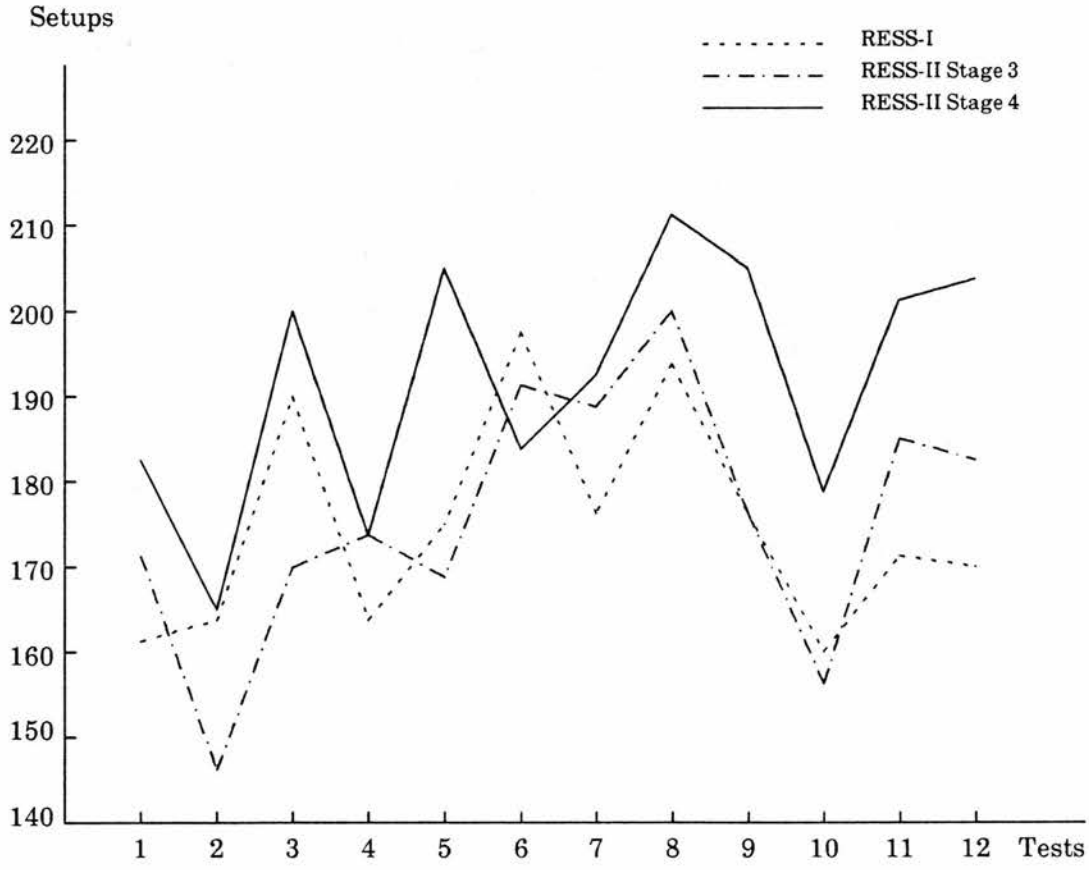


Figure 8.7 Number of set-ups saved over different tests (Environment three)

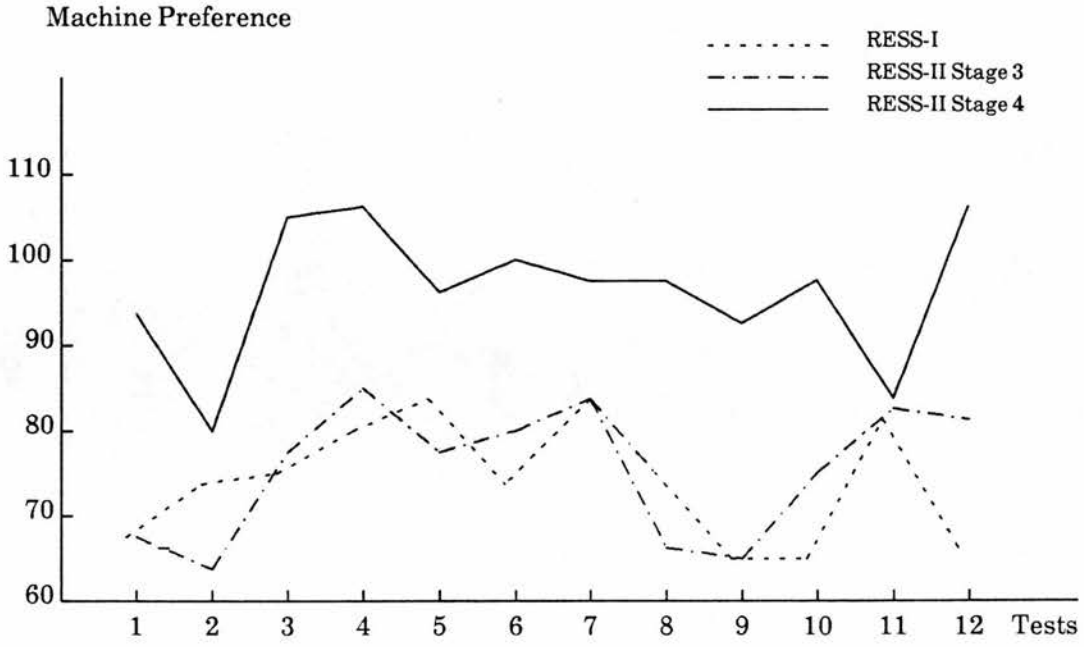


Figure 8.8 Number of machine preferences met at different stages over different tests (Environment one)

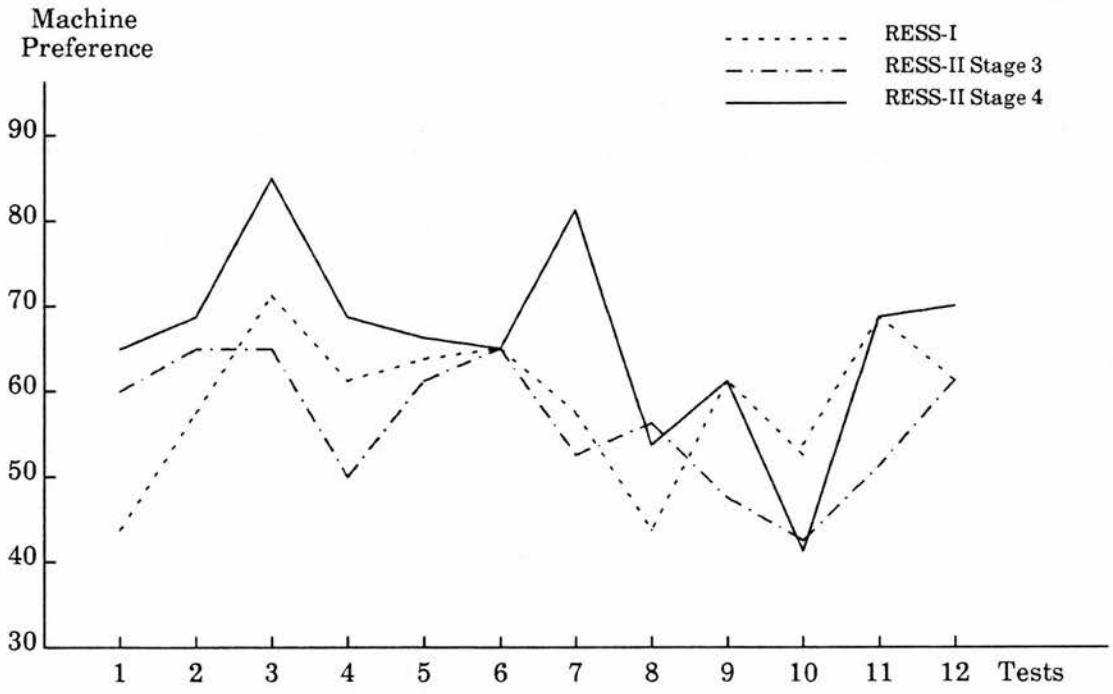


Figure 8.9 Number of machine preference met at different stages over different tests (Environment two)

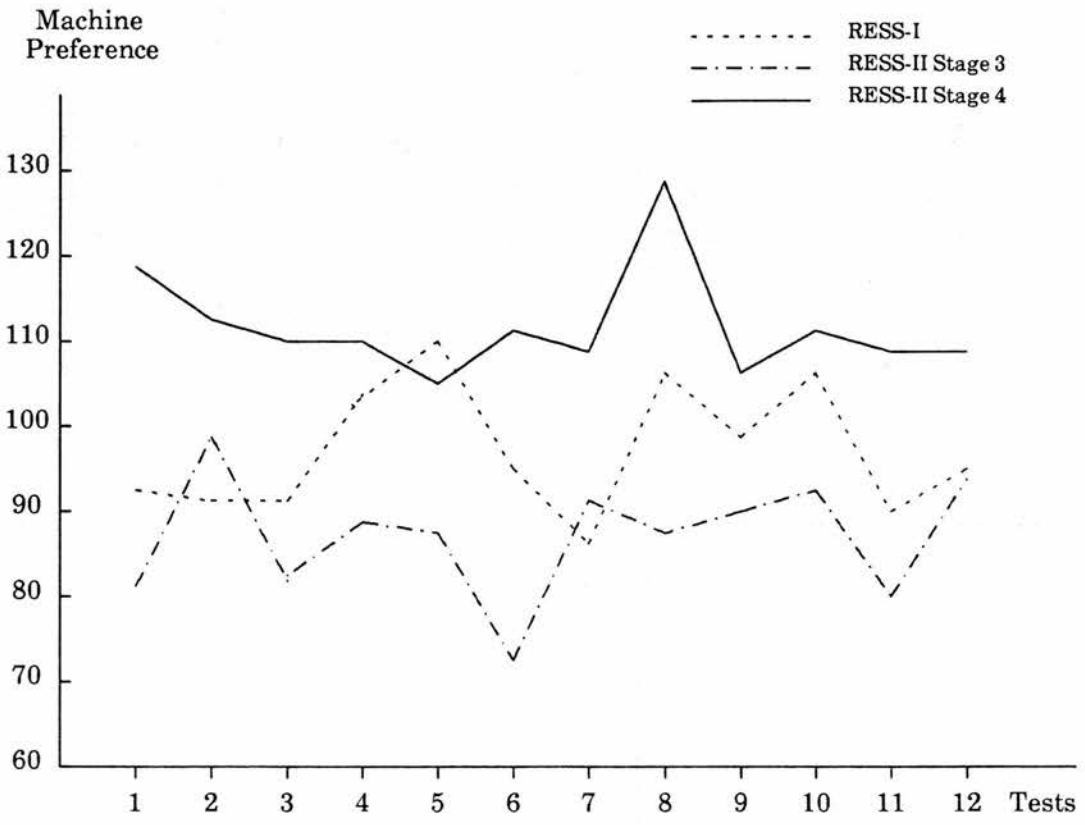


Figure 8.10 Number of machine preferences met at different stages over different tests (Environment three)

8.4 Summary

From the above figures and discussion, we can see that RESS-II shows schedule improvements over RESS-I, which indicates the effectiveness of the constraint decomposition approach. Furthermore, the results from RESS-II also illustrate the effectiveness of different stages in performing their tasks. Although some incidences occur, that is inevitable as scheduling is a highly context dependent and computationally NP-hard problem. Nevertheless, the experiment does suggest that the reinforcement and constraint decomposition approaches are very promising.

The shop environments used for our experiments are quite similar to real situations: the tests are conducted in continuous environments; the shop constraints and resources that have been considered include almost all the main ones in a real shop; only those specific ones, such as overtime and personnel requirement are not considered and they are unlikely to influence these approaches. Various measures used in the experiments to show the performance of the RESS-II system are adapted from [24] [25] [32] [45], however the measure of the number of machine preferences met over the different tests has been introduced for the purpose of this study.

The computational time for generating a single schedule is acceptably low. The average time required to produce a final schedule for a set of orders (around 140) through all those stages is one hour and 20 miutes in CPU time on the VAX11/750 using POP11 under POPLOG (version 9.2).

Chapter 9

A Comparison With Existing Work

This chapter summarizes the improvements that have been made over existing research and existing systems. It presents a comparison between existing work and the work that has been done in this research. Most of the points discussed below have already been touched upon in the previous chapters.

The discussion is partitioned into two parts. The first part concentrates on comparison between the algorithms (and features) of the RESS-II scheduling system and the past scheduling research in artificial intelligence.

The second part focuses on comparison between reinforcement planning and current AI planning techniques. It stresses the aspect of resource allocation and constraint satisfaction rather than the aspect of task ordering of AI planning. Two main techniques are compared with reinforcement planning and the constraint decomposition approach: hierarchical planning and backtracking. From these comparisons we can see that this research has made progress towards a general capability for resource handling and constraint satisfaction in planning as a whole.

9.1 A Comparison with Existing Scheduling Systems

In this section, we discuss the scheduling side of this research, in which RESS-II is compared with existing knowledge-based scheduling systems. Since most scheduling systems to date are fixed rule based programs as reviewed in chapter two, the comparison below is mainly with ISIS [28] [31] [33] and OPIS [32], which are two of the most sophisticated offerings currently available. The difference between these two systems and RESS-II is significant, ranging from the basic resource allocation technique to the scheduling methods they use.

The following lists the important points where we believe RESS-II [46] has not only made improvement over ISIS and OPIS, but also contributed to scheduling research.

(1) Resource allocation

Scheduling is a process of determining time bounds for each operation during which it should be processed, and of reserving the resources the operation needs. Its objective is to allocate limited shop resources to satisfy a large variety of shop constraints. Typically, these constraints conflict with each other. A successful schedule must reflect a good compromise between these competing concerns.

However, in order to solve the scheduling problem the most important thing is not constraint organization, i.e. how to do constraint generation, constraint posting and opportunistic constraint satisfaction. As far as we are concerned, as discussed in chapter three and four, it is fundamentally a resource allocation problem, that is how to best allocate available resources to operations in order to satisfy the constraints. Constraint organization is an inevitable part of the construction of a schedule anyway, but the underlying method (i.e. how to satisfy constraints or how to allocate resource) is far more important and it is the driving force for a good or bad schedule.

Although ISIS has a sophisticated modeling and constraint organization system, it does not have a strong method with which to tackle the problem of resource allocation and constraint satisfaction. Instead, it satisfies the constraints opportunistically and bases the scheduling decisions only on local analysis. During the scheduling of an order, it does consider the schedules of previous orders, but its beam search and constraint satisfaction rating mechanism do not consider what might happen in the future and what influences the current decision may have on the rest of the orders or even on the rest of the operations in the same order. A schedule decision can be highly rated, but it may cause great problems later. This deficiency also leads to other inabilities, such as allocating the less important resources.

In this research, conflict situations between resource availability and constraints' satisfaction and the conflict situations between constraints themselves are identified, and reinforcement scheduling and the constraint decomposition approach are introduced to deal with them. RESS-II schedules by analyzing the current situation and consulting some pre-built reinforcements (which act as an overall picture for the forthcoming events) such that each of the scheduling decisions is made with consideration of both overall as well as local situations. This is what is meant by having a global perspective.

Although ISIS and OPIS perform a post scheduling analysis to check whether the set of schedules constructed for an order is acceptable and if not backtracking is evoked to change them, such a change is still only based on local information and involves only local manipulation to the schedules of the operations in the order. The underlying idea is actually unchanged compared with the normal scheduling process before repairing (or backtracking). So, it does not solve the problem of looking ahead or predicting future effects either, and the process is still confined to this single order without any consideration of its possible interactions with the following orders. Typically, in such a situation, a problem with one order could be created by the schedules of previous orders,

and the schedules of this order could also create potential problems for the following orders.

In RESS-II, the situation is very different. By using reinforcements for detecting potential problems and resolving them in the scheduling process, the current scheduling decision will minimise the generation of undesirable effects on the following schedules. In such a process, backtracking will not be necessary as the current partial schedule situation and the possible future problems will have already been taken into account in decision making. Each schedule decision in RESS-II, therefore, makes a commitment, i.e. no matter what happens it will not be changed. This means that if the resource and constraint problems have been dealt with in the scheduling process, the result produced will be good. There is no need to evaluate such a result. Later on we will also explain why such an evaluation and then backtracking is inadequate for large problems involving resource allocation and constraint satisfaction.

A large amount of the work done by ISIS involves the extraction and organization of constraints that were created specifically for the problem being worked on. Only limited effort has been made to resolve such conflict situations. The current OPIS system [32] [33] (a direct descendant of ISIS), which attempts to make some progress by concentrating more on the heavily load work centre(s), does not yet have a good way to deal with the problems either; but they have been addressed. The central problem with these and other systems and approaches is that they all schedule according to local (or current) information and heuristics, and do not have a strong or positive method for good resource allocation and constraint satisfaction on an overall or global basis.

The same problem occurs in monitoring and rescheduling (or schedule revision) among existing systems as well. However, in RESS-II while maintaining as much of the old schedule as possible, the system employs the reinforcement scheduling approach so that possible future situations will also be taken into account in making rescheduling decisions. Furthermore, as far as the

rescheduling process is concerned, schedule revision in the ISIS and OPIS systems is done by locally changing the affected schedules of operations or rescheduling the affected orders, which are two different processes. In RESS-II, local updating goes hand in hand with rescheduling of the affected operations. In other words, when a problem is found there is only one revision process, i.e. if an operation can be kept in its original position it will be kept there and if not, another operation is selected to be scheduled in such a position. This gives a more uniform method of rescheduling.

(2) Scheduling method

ISIS uses a single-order scheduling method which takes an order consisting of several operations as the schedule unit. It has the defect of imposing an extra restriction on the scheduling process, specifically an order restriction which prohibits the consideration of operations of the other orders while scheduling the current one. In each scheduling cycle, a selected order has to be scheduled from its first operation to its last operation. This restriction results in creating more idle time on resources (both machines and tools), and the problem is compounded by the inability to change the schedules such that the productivity is decreased, and consequently some orders that could be finished on time are delayed. If there is only one machine in each work centre, the situation is worse. With more machines in each work centre there will be more choices. See chapter three for more details.

In the RESS-II system a parallel strategy is employed, which takes each individual operation as the basic scheduling unit and eliminates the order restriction. Instead of starting by choosing a single order to be scheduled as in ISIS, each scheduling cycle in RESS-II starts by finding a set of operations waiting to be scheduled on a selected machine (this machine defines a scheduling state along with its idle start time). These candidate operations are from different orders, and at this stage each of them is considered as an individual with no connection made to its order. However, during the decision

making, each operation will be regarded as an element of its order, which means the whole order will be considered as an entity. A decision is made with regard to the reinforcement analysis and the current situation analysis on the satisfaction of the constraints of the orders and as well as the operations. The removal of the extra restriction on the scheduling process should produce improved schedules and increase flexibility.

(3) The constraint decomposition approach

This research has also introduced a constraint decomposition technique for constraint satisfaction problems. From our experiments, it has shown great advantages.

In previous systems, only one scheduling process exists (e.g in ISIS and OPIS [33]) and all the constraints are considered in this single process (this is different from the levels in ISIS, which are just the scheduling steps in the same process), irrespective of type of constraints. These systems assume that through constraint satisfaction rating, a good compromise can be reached among these competing factors. However, constraints typically conflict with each other, so if a system does not know what the constraints' satisfaction situation is and where the conflicts between constraints may occur, it will have no grounds on which to arrive at a good compromise. As discussed in chapter three and chapter four, the compromise produced in such systems is not reliable at all.

Although all the constraints are also considered in a single scheduling process in RESS-II, good knowledge is available regarding constraint satisfaction and constraint interaction since reinforcement schedules have been built in a constraint decomposition fashion through several separate scheduling processes before the actual scheduling (or main scheduling). The possible conflict situations have been identified and an overall picture has been provided, such that the constraint satisfaction situation can be more precisely predicted in

decision making. Consequently the rating produced can represent a sound compromise among competing constraints.

Another point that needs to be mentioned is that without reinforcements to provide information about the conflict situations between constraints on the scheduling process, scheduling decision making may have to focus so much attention on the satisfaction of some important constraints that the less important constraints do not receive sufficient consideration. This is to say that less important constraints are very likely to be relaxed. Since the important constraints' satisfaction does not fix each operation's schedule to a specific position, typically there is a range where the constraints can be satisfied. Within this range, less important constraints may also be satisfied (or there may be choices where both kinds of constraints can be satisfied). As previous systems do not have a clear picture about constraints' relationships and how they are affected in the future, many less important constraints that could be satisfied are relaxed. This gives constraint decomposition an opportunity to claim one more great advantage, i.e. it exposes the relationship of constraints in a real schedule context, such that schedules may be manipulated in order to satisfy more less important constraints while also maintaining the satisfaction of the important ones.

(4) Constraint classification and satisfaction

Constraints and their satisfaction are confused in ISIS and OPIS. Although they do classify constraints as objective constraints, physical constraints, causal restrictions, availability constraints and preference constraints, their classification is only based on the organization aspect but not on the characteristic of constraint satisfaction or their different influences on the scheduling process and scheduling result.

RESS-II has taken a more comprehensive approach. Constraints are classified as preference constraints and technical constraints. Preference constraints are

further categorized as global constraints and local constraints, and technical constraints classed as ordering constraint and resource availability constraints. (They are explained in chapter four.) Preference constraints can be relaxed during the scheduling or planning process, while the ordering constraints and the resource availability constraints have to be satisfied. Hence, such a classification not only shows the different nature of constraints but also indicates how they should be considered in scheduling process. This sets up the base for a constraint decomposition approach.

An important thing about resource constraints is that although there is nothing that can be done about them as they must be satisfied in any situation, this does not mean they will not influence preference constraints' satisfaction. In fact, it is the constraints' satisfaction that provides information for the system to make decisions regarding why the resources should be allocated to one particular operation instead of another at a scheduling state (as there is typically more than one operation waiting to be scheduled). Each allocation of resources will typically influence subsequent allocation to the later operations. So, we can say that the allocation of resources will ultimately decide a schedule.

(5) Resources other than machines

Most of the previous systems do not consider resources other than machines. They simply assume that these resources will always be available when needed. But in reality, this is often not the case. These less important resources can also be overloaded in certain short time periods (as mentioned before), and such overloading also causes scheduling problems. Hence they must be considered if a good schedule is to be produced.

There could be two such situations. The first is when overloading does not happen at the same time periods as overloading of the important resources (the machines). In this situation, overloading can be balanced in the same way as balancing the important resource congestion. The second situation is when they

happen at the same time. In such a case, balancing the load on the less important resources will certainly have a positive effect on the important ones since any operation has to use an important resource (machine) anyway.

RESS-II has tackled this problem. Taking advantage of the reinforcements which also show the possible problems of the tools other than machines, it can bring tool resource problems into the decision making process so that these less important resources are also allocated to avoid congestion.

ISIS and OPIS also consider the tools in their scheduling process, but they only check their availability and reserve them for operations. They do not allocate these resources in order to avoid the possible high-contention periods.

(6) Search and search space

Search and search space are different in RESS-II from those in ISIS. As discussed in the previous chapters, scheduling is a NP-hard problem and the search space in a chronological backtracking or enumeration approach looking for the optimal result is enormous. So, practically, only a small part of the schedule space can be explored to produce a suboptimal or a feasible schedule. This will impose great difficulties on search and search control. Two basic questions that need to be answered are: which part of the search space should be searched to arrive at a feasible result and what are the evaluation criteria for differentiating the alternatives produced from such search.

In order to further discuss this issue, we would like to divide the search space into a breadth sub-space and a depth sub-space. Breadth sub-space represents the local search space, i.e. at any local scheduling/planning state all the alternatives that can be used. Depth sub-space expresses the situation where selection of one alternative instead of another during the search in breadth sub-space at a choice making point will produce a different breadth sub-space for the subsequent choice making.

In ISIS, the search methods involve the use of search space pruning heuristics and beam search, which map to the breadth sub-space of the partial schedule. It also does a limited depth sub-space search, using backtracking for schedule repair. However, this is far from being sufficient for making good decisions as discussed above in (1). In RESS-II, instead of searching the depth sub-space which is huge, the system searches some much smaller spaces created in place of the depth sub-space to reinforce the search in the breadth sub-space of the partial schedule. This results in less overall search. In fact, the search in those smaller spaces (pre-built reinforcement schedules) should be more appropriately called analysis process because it helps to foresee the potential problems. Hence, the search in partial schedule in RESS-II also only involves breadth subspace, but the analysis of reinforcement schedules provides good information in order for the system to solve problems in the equivalent depth sub-space. Such a concept emphasizes predicting the effect of current decisions and avoiding potential problems. If these problems can be solved, the constraints will be reasonably satisfied. As was stated before, RESS-II does not employ search space pruning or the dispatching rules paradigm, instead it looks at predicting problems and solving them.

RESS-II uses a parallel scheduling method to choose a machine (on which to schedule an operation) to start the search, while ISIS uses a single-order method. The search in ISIS is only in partial schedule space as it has no reinforcement schedules. The search in RESS-II involves both a partial schedule and reinforcement schedules. So, the partial schedule is a common search space. However, the partial schedule space in RESS-II is slightly different from the one in ISIS. In ISIS, it consists of states which represent the partial schedule, i.e. the current schedule situations of the machines, while the partial scheduling space in RESS-II also includes the operations that are waiting at the machine selected.

So, reinforcement scheduling involves search in several spaces, namely the partial schedule and the reinforcement schedules. These spaces look very big,

but for each decision only a small part of their spaces needs to be searched in order to get the information required by decision making. The current partial schedule, as well as the reinforcement schedules, are structured in periods (weeks) so that it is very easy for the system to track and analyse. Search in the partial schedule, which is the above mentioned breadth sub-space, simply accesses the current load on each machine. Search in the reinforcement schedules, which is an analysis process, looks for the possible future loading conditions of the work centre concerned and for long waiting operations of the order being worked on. The former only involves retrieval of the load and the latter only needs to check all the operations in the order. Both checks are relatively simple. These searches do not need to check what has been done and what has been left, as updating is performed to the reinforcement schedules after each schedule decision has been made, and this updating action is not difficult either. So, the search in reinforcement scheduling is quite simple. However, this does not necessarily mean that reinforcement scheduling can reduce running time a great deal. In fact, it will take a relatively long time because there are four complete scheduling processes to be run in order to produce a final schedule, although each of them are easy to compute individually.

As in ISIS, RESS-II also uses problem abstraction to provide strategic information to guide the problem-solving effort. Search spaces are gradually constrained. An example of this use of abstraction occurs in RESS-II; machines in the shop are grouped by function into aggregate resource units called work centres (machines in a work centre represent substitutable machines for performing an operation). Generally, in assessing loading conditions the system only checks the work centre that the current machine belongs to but not each individual machine. The resulting loading condition report will decide how the machine should be scheduled, and how the detailed problems should be tackled.

(7) Global constraints

In RESS-II, the global constraints are expressed by many other constraints and resource problems in the scheduling process because it is very difficult to consider them as individuals. For instance, there is no way to foresee precisely when a order will be finished at the scheduling of an earlier operation, especially, with so many orders to be scheduled. Furthermore, satisfaction of each order's constraints also has its relative priority among other orders. For example, the due date constraint of one order may be more important than the due date constraint of another order. Given these two situations, the due date constraint of an order may be expressed by both order importance and its dynamic urgency.

(8) Scheduling control

Many controls are necessary for a system dealing with a dynamic environment such as in job shop scheduling. During the schedule construction, satisfying all the constraints for the entire schedule is not the full story since execution exceptions will lead to schedule revision during the real production process. So, certain constraints, such as local preference constraints, only need to be positively considered for a short time period, while for the rest of the time they should only be considered arbitrarily.

Since schedule interruption happens all the time in the real manufacturing process, in revising the existing schedule only a part of the schedule needs to be updated starting from the time of interruption.

(9) Case study

Since the RESS-II system is not the result of a particular case study but built on the general job shop scheduling environment some specific problems are not considered, as in ISIS (e.g. shifts, overwork and physical constraints of machines

and products). Actually it is very difficult to consider physical constraints because in the general shop situation all sorts of products are manufactured and which have different shapes. In the ISIS environment, only certain kinds of products are produced, turbine blades of different sizes. The representation is much easier.

9.2 A Comparison with the Current Planning Techniques

In this section, we compare reinforcement planning and the constraint decomposition approach with existing AI planning techniques, and we will describe progress that has been made as a result of this research. The contribution is mainly on resource allocation and constraint satisfaction aspects rather than planning paradigms or plan organization aspects. Essentially, reinforcements are external aids to planning/scheduling decision making. This technique is not only useful to resource allocation and constraint satisfaction, but may also be helpful to situations not involving resources.

9.2.1 Resource Allocation and Constraint Satisfaction

(1) Resource allocation in planning

In most realistic planning domains as well as scheduling domains, ordering of actions is only one of the problems. Resources and constraints also play an important part, sometimes even a greater part, than action orderings. In such domains, good resource allocation and constraint satisfaction is often required. However, current AI planning research is just beginning to tackle the problem [8]. The central idea is still focussed around arranging the ordering relations by detecting and correcting interactions. They often implicitly assume unlimited availability of resources, or simply check their partial solutions against the

actual resources available. However with constraints to be satisfied, resource allocation will not just be a problem of checking the ordering solution against the resources available — it is much more than that. The central point is how to allocate limited resources to satisfy various constraints. As resource problems are generally NP-hard, trying every possibility to find the optimal solution may work in small problems, but for large problems it is impossible to depend on uninformed search. Decision making about constraint satisfaction has to be introduced.

Even without resources and constraints, at any planning state there could be more than one choice or alternative choices that can develop the plan/schedule further. However, many of these choices may not lead to the goal state. So, a decision has to be made to choose a good one to develop the plan. Most of the time, the system may not choose the right one the first time because there is insufficient information to differentiate between these alternative choices. If something goes wrong later, typically backtracking is performed to consider alternative solution paths. For large problems with resources and constraints, such backtracking is impractical and even unhelpful (this will be further discussed later in this section). Decision making then becomes more acute.

Although past research has developed many techniques for decision making, mostly, they only rely on heuristic rules or local situations but do not give sufficient consideration to each particular problem and the overall situation. For most realistic tasks, this is insufficient because good decision making requires reliable information both about the present and the future. For this purpose, the reinforcement approach may provide a promising approach.

(2) Distinguish plan influences

In every domain, there are typically many factors that influence planning, such as resource availability, action ordering, planning objectives, etc. Current planning does not distinguish between these factors but simply calls all of them

planning constraints. This is quite inadequate because their functions and their influences on a planning or scheduling process are very different. As discussed in chapters one and four, ordering forces one action to be performed before and/or after other actions, while resource availability puts an extra restriction on the execution of actions. The objective constraints are normally meta-constraints which express a desired final planning result.

Due to resource availability restriction, sequencing links between actions may need to be introduced in the planning process. These links will not normally affect the legitimacy of a plan, but can be very important for constraint satisfaction. For example, in scheduling, most of the links between operations are sequencing links (ordering links also exists, describing the precedence of operations), and it is this sequence of operations that decides the final schedule and its constraint satisfaction result.

The resource handling capability of current domain-independent AI planning is still very limited. Only consumable resources can be considered [8]. Shared resource handling is still outside the reach of most planners. The reason could be that such planning capability takes the form of adding to the existing set of ordering constraints which can be represented without resorting to sequencing links and disjunction. However, handling of shared resources requires extensive use of sequencing links and disjunction in representing the possible sequence of activities.

(3) Two ways of planning

Perhaps there are two ways to do planning or scheduling involving resources. One is to first plan the actions and then check the resource availability, changing plan resource constraints accordingly, as in O-Plan [8] [10]. In such planning, actions and their orderings constitute the basis for plan representation. The other way is to find resources first before choosing an action to allocate the resource to, such as in RESS-II. In this situation, resource

availability is the core for plan/schedule representation, which means that resource availability is checked before actions and orderings are considered. The former emphasizes actions and their orderings while the latter stresses that without resource no action can be performed (we call this a resource oriented approach).

What are the advantages and disadvantages of these two approaches? We do not have a clear assessment, but they seem to suit different situations. Most current AI planners use a least-commitment approach because the essential part of their planning processes is about actions and their orderings (not sequences). RESS-II employs resource oriented planning due to the fact that the orderings among operations are not very complicated in the scheduling domain but sequencing and temporal relations are very important. If in some other domains where resources and orderings are both extensive, further research will be required. Perhaps integration is the answer.

9.2.2 Reinforcement Planning and Hierarchical Planning

A hierarchical planning approach generates a hierarchy of representations of a plan in which the highest level is the simplest or the most abstracted level, and the lowest level the most detailed level. They are levels of abstraction. It tackles the problems by a divide and conquer strategy; breaking a complex problem into subproblems which form a plan down to some level of detail and then refining this high level plan to more detail until the whole plan had been refined to a complete sequence of detailed primitive actions. The subproblems can also be divided in the same way. The advantage is that the plan process is very clear with different levels of activities and different aspects of a problem being tackled in different parts so that the computation will not be overwhelming, with endless backtracking and hence difficult to manage.

Many systems have implemented this approach. In fact, it has become a major problem solving and planning technique. ABSTRIPS by Sacerdoti [47] is the earliest such

system. It is a hierarchical version of the STRIPS [2] system, which plans in a hierarchy of abstraction space. By adding abstraction reasoning, the performance of ABSTRIPS was much superior to that of STRIPS. Subsequently, NOAH [6], NONLIN [7], and MOLGEN [12] all adopted this approach.

However, reinforcement planning is quite different. It does not affect the main planning paradigm and planning structure of a scheduler or planner. Instead, it helps with decision making in the planning/scheduling process. In other words, planning/scheduling strategy and organization in reinforcement planning could be the same as normal planning, but its decision making process will be much more sophisticated and reliable, because reinforcement plans/schedules are generated and used to show potential problems and to provide information for the main scheduling process. For instance, in RESS-II, stage three is the main scheduling process, while the previous scheduling stages are all reinforcement building stages. Each stage is a separate planning process just like the normal scheduling, but for exposing different scheduling problems.

From a planning organization point of view, hierarchical planning is a top-down planning paradigm or a systematic way of problem solving. It can be characterized by a layered structure, which emphasizes how a planning process should be managed. But its layered structure is only within a single planning process, and does not concern itself with how a decision should be made at a choice point. Reinforcement planning; on the other hand, not only involves different planning/scheduling processes, but also uses the results from these processes for its decision making. Each of the reinforcement building processes is created for different use, and each of them may employ hierarchical planning techniques or other paradigms for its planning organization. There is also only one main scheduling process in this approach. As this is a separate planning process too, it can be organized with any planning paradigm.

In some cases, hierarchical planning can also be interpreted as creating plan islands to be tackled separately and then linked together. The different planning/scheduling processes in reinforcement planning, concentrate on different conflict situations and can

themselves look like plan islands to be tackled, but they are not actually plan islands as they provide an overall picture of the conflict situations. In other words, they are problem manifestation processes rather than problem solving processes. Dealing with these problems is still the task of the main scheduling process.

We may also notice that the reinforcement building stages in RESS-II use abstracted knowledge of the domain while neglecting the detailed information. For instance, at the first stage of RESS-II only work centre and due date constraint are chosen to be considered for building the first reinforcement schedule. This is not to create an abstraction level, but instead it focuses on a specific conflict situation, and the individual machines and other constraints will not have much influence here.

In order to make a decision the reinforcement planning process will not only consider local situation and heuristic rules, but also analyse reinforcement plans/schedules. After a decision has been made, it will also need to update these external reinforcement plans/schedules.

In the above, we said that reinforcements are for decision making. However, what does decision making include? As far as reinforcement planning is concerned, decision making is not limited to the action level, but also includes the meta-level. In such a context, reinforcements also provide information to guide the planning process and the use of constraints. For instance, in RESS-II, when the reinforcement analysis reports that the machine to be scheduled is in a bottleneck period, then the operations waiting to be scheduled will be evaluated in a different way from the normal situation where the machine is not overloaded. Different sets of constraints may also be used and different weights attached to each constraint's satisfaction.

9.2.3 Reinforcement Planning and Backtracking

In many situations, it might be very difficult to choose the right path which will lead to the goal state the first time so the system reaches a dead end or unsatisfactory situation.

Then the system must go back to previous plan states to choose an alternative route to continue the planning process hopefully to a feasible solution. There are two main techniques for backtracking. One is chronological backtracking and the other is dependency-directed backtracking.

In chronological backtracking, the system saves the alternative paths at every choice state encountered. If the chosen path cannot reach the goal state specified, it backtracks to the most recent choice state first and chooses another path from the set of alternatives at this state. If none of these alternatives can lead to the goal state, the next most recent choice point is tried until a suitable path or plan is found. The computational effort is exponential and for many problems it is impractical to use. Earlier systems used such an approach as their means to control search. The PROLOG programming language is a good example.

Some systems do not keep saved states of the partial solution at choice point. Instead they record the dependencies between decisions, the assumptions on which they are based and the alternatives from which a selection can be made. When a failure occurs, they first analyse the failure to find out the reason, and then track back to the relevant choice point by using the recordings kept. Then they make another choice to avoid the mistakes in the previous process.

The basic concept of backtracking is that when problems are found late in the planning process, the system goes back to some choice point to select an alternative. The underlying concept of the reinforcement approach and the constraint decomposition approach is that the system should have reliable information concerning the resource and constraint problems before it makes any compromising scheduling decision, and some reinforcement schedules are built and used to provide such information. In other words, since it is very difficult to predict the conflict situations among various resources and constraints in a scheduling/planning process, pre-scheduling processes are conducted to build some utilisation plans to act as overall picture of the forthcoming events. Through analysing these plans, the system will be able to foresee potential problems so that a good choice can be made the first time without resorting to

backtracking. Using this concept, solving resource allocation and constraint satisfaction problems are substituted by resolving the conflict situations between resources and constraints, and between constraints themselves.

If this scheduling process is formalised as a backtracking problem, the first thing to do is to build a schedule, possibly with many heuristic rules. After a schedule has been built, the system analyses it to see whether it is acceptable by using some kind of evaluation criteria. If it is not, the system will go back to the schedule to change it.

Theoretically, chronological backtracking [1] [2] can find the best result. But when the problem and number of alternative choices grow large, it will be impossible for such backtracking to manage. Dependency directed backtracking [41] [42] is a more advanced method for choice making. As it works on logical dependencies between actions, it would not be very useful at decision making either as the choices left from the ordering scrutiny are beyond action dependency (as there is no strong causal relations between operations) or there is no clear dependency that exists. When problems occur, it is very difficult to find which decision(s) is/are responsible as all the schedules of operations are only loosely inter-connected by temporal constraints. The suspect choices may not cause the problems and the schedules of operations which have no problems could be the culprits as they may have adverse effect on the scheduling of the other operations. Generally, when a problem occurs, it is not normally some particular choice(s) that is/are to blame but rather a problem situation that was not properly tackled.

Even if some problems can be found with dependency-directed backtracking it may be pointless to make any change inside a plan/schedule. The reason is that planning/scheduling considerations normally involve more than one resource and more than one constraint. The changes made to one resource or constraint will typically have some influences on the others and possibly make them invalid. This kind of blind search may properly repair one problems but cause others.

For example, if the due date constraint of an order is not satisfied, then the system traces back and finds that one of its operation had been waiting for too long in the queue before it was scheduled. The system may try to shift this operation back to a earlier time. But this shifting will force some other operations scheduled on this machine to be pushed forward to a later time. This may cause the due date constraints of the orders of these operations to be violated. In such a situation, dependency-directed backtracking will not know what to do. The reason why this operation was in the queue for a long time may have been because of bad choices made earlier about the operations' schedules in this order and in other orders. In other words, it may have been caused by bad schedules earlier on the machines of the work centre concerned and other work centres. It is extremely difficult to find the exact causes. In reinforcement scheduling, instead of going back, the system goes forward to detect problems for the current decision making so that potential problems can be avoided or their effects be best reduced.

Consider also the setting up of evaluation criteria, or how the system can know in what kind of situation what plan/schedule is good and what is bad. Even if such criteria can be established, reliability is still a issue. A schedule which meets the criteria may not be a good schedule and one that does not meet the criteria may not be a bad one either, because a good schedule result depends on individual circumstances and it is extremely difficult to find a correspondence between the schedule situation and the acceptable result in such a situation. In reinforcement scheduling, however, a completely different methodology is used. It advocates identifying the conflict situations, and believes if these resource problems and constraint problems can be dealt with properly, a guaranteed good result will be produced. So, there is no need to check if the scheduling result is acceptable or not.

Although the proceeding discussion argues for the proposed reinforcement planning and constraint decomposition approach in favour of backtracking, we would not reject either one in favour of the other. Obviously, both models have merit and are suited to different situations. Reinforcement planning depends on predicting conflict situations and resolving them, and the result it looks for is a good one rather than the best one. So, if a problem strictly requires the best result or a result that met the criteria given, this

approach will not be appropriate. In such situations, perhaps only chronological backtracking can guarantee the result required (dependency-directed backtracking will not work either because it is directed by specific problems and it cannot deal with context dependent situations), although the computation is likely to be combinatorially explosive. However, as mentioned before, in most realistic domains the best result is practically impossible to obtain by enumerating (or chronological backtracking) all the alternative choices and a good result is often required. Then the reinforcement approach will be very useful.

Chapter 10

Summary and Conclusion

The central topics of this research are resource allocation and constraint satisfaction (in fact, they should be called one problem as the allocation of resources is for the satisfaction of constraints). In terms of Operational Research, most tasks that involve resources and constraints are computationally NP-hard. Past work in the field has suggested that there is no simple way to deal with them. In terms of AI planning, they are also difficult problems. In fact, past AI planning research has concentrated mainly on the ordering aspect of the problem, i.e. detecting and correcting interactions. Most systems do not handle resources and those that do only check the resource availability against the partial solution. However, the underlying problem of resource handling requires much more than simple checking. With constraints to be satisfied, the planning or scheduling result required will need to be efficient. It is actually an organizational problem that concerns both current decisions and the future impact of such decisions. This is quite different from satisfying ordering requirements between actions and it forms another aspect of planning or problem solving, i.e. resource constrained decision making. Furthermore, the resource and constraint problem is also very important since in the real world there are few situations that do not involve resources and constraints. As planning becomes more realistic, resource handling becomes very acute.

This research has focused on these issues and tried to propose a way to deal with the problems. However, the effort is not one of how to take into account all kinds of resources and constraints, but how to allocate the resources such that the actions are well organized and the constraints are best satisfied, i.e. how to do resource allocation and how to positively satisfy the constraints. This is more than the ordering of actions in traditional planning as this activity is only performed after the ordering relations between actions have been established (although after a decision has been made about resource allocation it will also feed back to the ordering manipulation).

The domain in which this research has been carried out is job shop scheduling — an outstanding problem of resource allocation and constraint satisfaction. In the process, the scheduling task is extensively studied. We first identified the problems of resources and constraints: in particular, the conflict situations between resources and constraints and the conflict situations among constraint themselves. We then proposed ways to deal with them, i.e. reinforcement planning and the constraint decomposition approach.

10.1 Summary of this Research

10.1.1 Scheduling Problems

To analyse the problem, in this research we used the parallel scheduling method to illustrate the basic scheduling process, although the underlying problems of single order scheduling are the same. Choice making among alternatives in such a process is separated into two levels, the choice generation level and the decision making level. The choice generation level satisfies the ordering constraints (or precedence constraints) of each operation to produce a set of schedulable operations that can be performed on a selected machine. The decision making level chooses a single operation from this set to be scheduled on the machine (a machine can only take one operation at a time). We call this resource constrained decision making.

Since the objective of scheduling is to satisfy various constraints, so the criterion for making such a decision will be the satisfaction of constraints. However, many constraints, such as due date and work-in-process, express preferred final results and many problems only arise after some or all the scheduling has been done. It is almost impossible to access their satisfaction situations at a local state.

In fact, scheduling is a resource intensive task where every operation has to be carried out on a specific machine and possibly with the presence of certain tools. In such an environment, there are typically many orders to be scheduled and many constraints to be satisfied. If the resources have sufficient capacities, these orders will be scheduled satisfactorily and the constraints will be best satisfied. However, the resource capacities are limited over certain time periods, and if a resource is occupied by an operation it cannot be used by any other operation. So, orders have to compete with each other for the same resources and this causes problems.

However, not all the resources have the same effects on the scheduling. Some of them will not influence the schedule much as their capacities are well above the required capacities (i.e. they are underloaded machines), while some others will influence a schedule a great deal since their capacities cannot meet the demand of the operations so that some orders may be delayed (i.e. they are bottleneck machines).

So, the heart of the problem is that the limited resources cause constraint competition. There are typically conflict situations (the bottlenecks) which exist between constraint satisfaction and resource availability that restrict satisfaction of constraints.

Based on the existing resource capacities, however, the constraints will also conflict with each other. These conflict situations also hamper the satisfaction of one or other constraint. So, in order to achieve good constraint satisfaction these conflict situations (including the conflict situations between constraint satisfaction and resource availability) should be avoided or their effects should be maximally reduced. Hence, at this decision making level, the system must try to choose an operation to allocate to the resources in such a way that it will help to lead the scheduling towards a conflict free

situation. Then, the system needs to effectively predict when and where these possible conflict situations may occur. In other words, it needs to predict what impact the decision currently being made will have on the following decisions. Therefore, besides considering the present situation, this decision making level will also have to take into account the overall situation.

10.1.2 Reinforcement Scheduling and the Constraint Decomposition Approach

As in scheduling, it is the resource capacities that restrict constraint satisfaction, so satisfying constraints can be interpreted as solving resource problems. The approach advocated in this thesis assumes that if the resource problems can be tackled the constraints will be well satisfied. Hence, the focus of the decision making or choice making is on predicting possible resource problems and trying to avoid them or reduce the effect of them. It has two sub-issues. The first one is the local evaluation of possible resource allocation, which includes the current loading analysis and local consideration of constraints of each operation and the shop. The second concerns the global effects of such possible resource allocation. This offers ability to differentiate amongst the alternative sets of scheduling decisions based on the prediction of the future impacts of such decisions.

To deal with the problems, in this research, two techniques are introduced, reinforcement scheduling and the constraint decomposition approach. Although separate techniques they combine to form the basis of the approach taken in this thesis.

Since the central problem in scheduling is how to predict the effects of the local decisions on the global solution, we suggest that one or more pre-scheduling stages should build a global view of the forthcoming events (the reinforcement schedule), serving as a basis for generating decision-making guidelines for constructing the detailed schedule. The detailed scheduling stage schedules again by using these schedules as its reinforcements to help it to predict the future problems and to try to avoid or reduce the

effects of these resource and constraint problems. In such a fashion, the scheduling is a process of repeatedly analysing the current situation by searching through the partial detailed schedule and consulting the reinforcement schedules to make decisions. The search through the partial detailed schedule gives a current situation report on various resources. The search in the reinforcement schedule finds out if any possible problems exist both now and in the future and sees whether they can be solved by proper allocation of resources to the operations. Associated with these searches is an evaluation mechanism to make the final decisions. After a particular decision is made and resources have been allocated to the selected operation, the system updates the reinforcement schedule to make it also reflect the current detailed scheduling situation and make it more reliable for future analysis.

As there are many constraints that need to be considered in the scheduling process, we argue that it is very ineffective to consider all of them in a single scheduling process. The detailed analysis of the situation suggests that the constraint decomposition approach is feasible, which is characterized by considering different problems in different scheduling processes. It is a technique following from the reinforcement approach, without which the decomposition will not be possible as the inter-stage analysis and communication play an important role, not only in providing the problems but also in helping to maintain the previous constraint satisfaction results.

The reinforcement scheduling approach stresses that the scheduling process needs some external information which can provide an overall picture of the coming events to reinforce its decision making, and this outside information can be obtained from several scheduling processes. The constraint decomposition approach stresses that a single reinforcement schedule is not sufficient to expose various problems and it suggests that different reinforcement schedules should be built to show different conflict situations. These two approaches help each other.

Not only will the main scheduling use the reinforcement approach, but other processes can use it too, such as the second stage of reinforcement building, which uses the capacity plan from the first reinforcement building stage as its reinforcement schedule.

These two techniques simply help with the decision making during the scheduling process. They do not influence the normal planning paradigm but offer outside help. The scheduling process at each stage can also be done in a hierarchical fashion or any other way.

10.1.3 The RESS-II Scheduling System

Based on the approaches described above, a scheduling system called RESS has been implemented. Two versions have been built to date, RESS-I and RESS-II. This thesis concentrates on a discussion of RESS-II. The scheduling environment it uses has been described in chapter two. The major difference between the two versions of the system is that RESS-I did not implement the constraint decomposition approach so that all the constraints were considered in one scheduling process. The conflict situations among constraints themselves were therefore not well handled and as a result RESS-I produces poorer schedules compared with RESS-II.

The scheduling in the RESS-II system consists of four parts: reinforcement building, schedule construction, schedule refinement and rescheduling. Each part may have one or more stages of scheduling.

(1) Reinforcement building

The reinforcement building has two stages of scheduling. The first stage builds a capacity plan to show the problems between the resource capacity required and resource capacity available in order to meet the due dates of the orders. As discussed in chapter 3, this process will not consider much detailed information, so it is not a schedule that can be directly used in a real factory.

The second stage tries to solve those resource problems (the conflict situations between the due date constraint and resource availability) in order to satisfy the due date constraint, and also shows the problems between the due date

constraint and the work-in-process constraint. The capacity plan generated at the last stage is used as the reinforcement.

(2) The main scheduling

This part is a single scheduling process. After the last two stages of reinforcement building, the system has gathered sufficient information about the resource and constraint problems, this scheduling process can generate the real schedule for use. The reinforcement scheduling approach is also used, and both of the reinforcement schedules are employed. This is the main scheduling process.

To some extent, this process can also be interpreted as trying to reduce the work-in-process time, and at the same time maintaining the due date constraint satisfaction achieved at the last stage (the second stage of reinforcement building).

(3) Schedule refinement

The schedule refinement part also has only one stage. It refines the schedule constructed in order to satisfy as many local preference constraints as possible while maintaining the previous constraint satisfaction result. It uses the scheduling result from the main scheduling stage as the background, against which the operations are shifted around on the same machine, or on different machines in the same work centre, to try to satisfy more local preference constraints without violating any global constraints.

(4) Monitoring and rescheduling

The rescheduling also has only one stage. Its function is to deal with the unexpected situations and to maintain the shop stability. Reinforcement scheduling is again used, and two reinforcements are employed: the first stage

result and the final (existing) schedule. Unlike the scheduling construction, this rescheduling stage will not use the constraint decomposition approach. So, all the constraints and interruptions are considered in a single scheduling process.

From the above brief review, we can see that although the use of the reinforcement approach at different stages in RESS-II is slightly different, this technique provides a uniform approach to the scheduling problem from the construction of a schedule through to production monitoring and rescheduling. The particular implementation of this technique in the RESS-II system is domain dependent. There could be some other ways to implement it.

Many tests have been carried out on the system, with different numbers of work centres, different numbers of machines in each work centre, and different numbers of orders and their starting times and due times. The results as presented in chapter 8, suggest that this is a promising approach.

10.2 Limitations and Possible Future Work

(1) Constraints Representation needs to be further refined and extended

In RESS-II, the constraint representation is not very sophisticated since the main interest of this research is resource allocation and constraint satisfaction. Many alternatives should be represented. For example, a due date constraint could be specified as a series of acceptable dates spanning a period of time, and each of these alternatives could also be assigned utility information to specify which are the preferred ones. Such data would provide the system with constraint relaxation information. Furthermore, in real situations, there could be many aspects of a constraint that need to be considered, but in RESS-II, only one value is provided for each constraint.

Some extension could also be made in quantifying the notion of constraint satisfaction so that ratings would be different according to different degrees of satisfaction. In RESS-II, although the rating changes over different situations, within the same situation the degree of satisfaction is not considered. This may be inadequate as different degrees of satisfaction could have different influences on the scheduling process. However, it is very difficult to access, although it may not be very difficult to include a mechanism to take it into account. Further research is needed.

(2) Compromising decision on strict constraint conflicts

The constraint decomposition approach in this research can only be used in situations where the constraints concerned are nearly independent, e.g. due date and work-in-process. This enables a schedule to be changed in order to satisfy later constraints without having too much influence on the final result of earlier constraint satisfaction. When later constraints do conflict with earlier ones, the later ones have to be relaxed, therefore there is no compromise decision made. This is the policy of the constraint decomposition approach in RESS-II. For example, in a situation where a decision has to be made as to whether an idle time period should be taken, a positive decision may help the due date constraint satisfaction and increase the resource usage, but this may also increase the work-in-process time. In such a case, RESS-II cannot make any judgement as to which constraint to relax, but simply relaxes the work-in-process constraint.

(3) Tailoring the system to make it useful

Since this system was built without real factory data, it may not be suitable for a real situation. Although the main requirements and constraints have been considered, there are still some things that have not been taken into account, such as overtime, shifts, alternative process routings, etc. Furthermore, different real situations may have their specific constraints and criteria to

evaluate the constraints' satisfaction (for example, the inventory should be kept to a certain level). Further investigation of the application of the RESS approach to the real world is required.

(4) Integration with other management functions

Scheduling is only one of the functions in production management. To link it with the other functions is a natural research direction. An obvious example is the process planning function, which prescribes the process routings for the products. During the scheduling process, if a routing is not suitable for current situation (it may use a bottleneck machine), the process planning function could be invoked to find an alternative routing. At the moment, RESS-II only uses the fixed routings, so when problems are encountered the only alternative open to the system is to best reduce the effects of the problem according to the current situation.

(5) How to consider other kinds of resources

In our scheduling situation only shared resources are involved, so consumable resources and other kinds of resources are not handled in this research. Since the nature of different resource types are quite different, their consideration will also be quite different. For example, use of consumable resources has no time dependency, only a quantity relationship. In order to achieve the objective of handling consumable resources further research will be required. This is so for many other types of resources. If various constraints are involved in a single domain, the problem will become even greater.

(6) New ways to implement the reinforcement approach

As mentioned in chapter three, there could be many ways to implement a reinforcement approach to scheduling. RESS only represents a single application of the principle.

There is a limitation to the current implementation, i.e. its first reinforcement building stage can only be used for a large environment where many orders are involved. It will be quite difficult to use this for small problems with only a few orders involved. The reason is that the load in this reinforcement scheduling is organised into time periods (the current unit is a week) and the capacity analysis in the second and the third stage of scheduling is also based on time periods. Although these time periods can be changed to longer or shorter times, it is not very flexible because for small problems such a loading representation is very difficult to arrange. Even if it could be done, the reliability is questionable. Further, if the load times of operations vary a great deal, the prospects for this approach deteriorate.

Other ways of building the reinforcement schedule may not have this problem and could be more flexible because in such implementations the reinforcement schedules could take the same form as the main schedule and the analysis could be done without being restricted to certain time periods. We will consider such implementations sometime in the future (the one mentioned in chapter three could be one, for instance).

(7) How can the reinforcement approach be used in other domains

The approaches introduced in this research are not aimed at achieving optimal solutions to the resource allocation problems. Instead, they try to produce a good (i.e. feasible) result by predicting and resolving the resource and constraint problems.

Like scheduling, in many other realistic resource intensive domains, there are always some outstanding problems with certain resources that influence a plan or schedule a great deal. So predicting and trying to prevent such problems from happening or to reduce their effects at an early stage will certainly help to produce a good result. In such cases, the reinforcement approach will certainly be helpful.

However, for the problems like "travelling salesman", the plan result is completely context dependent, i.e. any change to the plan will have unpredictable effects. Reinforcement planning would do no good here.

(8) Some speculations about AI Planning.

Although this research is mainly about resource allocation and constraint satisfaction in scheduling, it also sparks off some speculation about AI planning.

Current AI planning is still largely confined to detecting and correcting ordering interactions between actions. The resource handling capability is still very limited. Normally, in resource-restricted domains, the total amount of resources available and/or the total capacities resources have over a time period are limited. The actions that meet the ordering requirement may be restricted in their execution by limited resource. For example, two actions that can be performed in parallel may not be executed in parallel because there is only one unit of resource available. So, one has to be executed after the other. This will introduce a sequencing link between these two actions (apart from the ordering links with other actions in the plan). So, there should be two kinds of links that need to be considered in planning: ordering links and sequencing links.

Although both of these types of links express a precedence that one action should be executed before, after or at the same time with other actions, their underlying meanings are quite different. The sequencing links are not as rigid as the ordering links, which means that the sequence of actions that are caused by resource limitation can be changed. In other words, rearrangement of the sequencing links does not affect a plan's legitimacy if the ordering links are still there. The combination of these two will certainly make planning more powerful (but also more complicated). Perhaps for such a combination, the network representation of plans should be integrated with the resource-based representation of schedules as discussed in this research, where the time and the resource are automatically included.

Generally, as mentioned before, planning may need to be considered as two separated aspects, ordering aspect and decision aspect. Most of the work done to date is either concerned with the ordering aspect, such as NONLIN [7] and NOAH [6] or with the decision aspect, such as RESS-II [46], ISIS [28] and OPIS [32] (there is no great demand for the ordering aspect in scheduling). There is still no system available that can address both sides of the planning problem.

It is also possible to implement the reinforcement approach in a domain independent planner as this technique does not affect the main planning paradigm. The reinforcements are simply outside help to show the internal relationship of the resources and constraints for planning decision making. However, the implementation of this technique is likely to be domain dependent rather than domain independent as different domains may have their own ways of building and analysing reinforcement plans.

References

[1] Ernst, G.W. and Newell, A., GPS: A Case Study in Generality and Problem Solving, Academic Press, New York, 1969.

[2] Fike, R.E. & Nilsson, N.J., "Strips: a New Approach to the Application of Theorem Proving to Problem Solving," Advanced Papers of International Joint Conference on Artificial Intelligence, 1971, pp.189-208.

[3] Sussman, G.A., A Computational Model of Skill Acquisition, American Elsevirer, New York, 1975.

[4] Tate, A., "Interacting Goals and Their Use," Proceedings of Fourth International Joint Conference on Artificial Intelligence, 1975, pp. 215-218. Tbilisi, USSR.

[5] Waldinger, R., "Achieving Several Goals Simultaneously," SRI AI Centre Technical Note 107, SRI, Menlo Park, CA, USA.

[6] Sacerdoti, E.D., A Structure for Plans and Behavior, North-Holland, New York, NY, 1977.

[7] Tate, A., "Generating Project Networks," Proceedings of Fifth International Joint Conference on Artificial Intelligence, 1977, pp.888-893, Cambridge, Mass. USA.

[8] Bell, C., Currie, K. and Tate, A., "Time Window and Resource Usage in O-Plan," AIAI-TR-32, University of Edinburgh, 1987.

[9] Vere, S.A., "Planning in Time: Window and Durations for Activities and goals," IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI-5/3, 1983, pp. 246-267

[10] Currie, K. and Tate, A., "O-Plan — Control in the Open Planning Architecture," Expert Systems 85, ed. Martin Merry, Cambridge University Press, 1985, pp. 225-240.

[11] Miller, David P., Planning by Search Through Simulations, PhD Thesis, Yale University, 1985.

[12] Stefik, M.J., "Planning with Constraints," Artificial Intelligence 16, 1981, pp.111-140.

[13] Stefik, M.J., "Planning and Meta-Planning," Artificial Intelligence 16, 1981, pp.141-169.

[14] Hayes-Roth, B. and Hayes-Roth, F., "A Cognitive Model of Planning," Cognitive Science 3, 1979, pp.275-310.

[15] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R. "The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty," Computing Surveys 12(2), 1980. pp.349-389.

[16] Hayes-Roth, B. A Blackboard Architecture for Control, Artificial Intelligence, 26(3). 1985, pp.251-321.

[17] Wilkins, D.E., "Representation in a Domain-Independent Planner," Proceedings of Eighth International Conference on Artificial Intelligence, 1983, pp, 733-740, Karlsruhe, West Germany.

[18] Wilkins, D.E., "Recovering from Execution Errors in SIPE," Computional Intelligence 1(1), 1985, pp33-45.

[19] Smith, S.F., "A Constraint-Based Framework for Reactive Management of Factory Schedules," Proceedings International Conference on Expert Systems and the Leading Edge in Production PLanning and Control, 1987.

[20] Collinot, A., and Le Pape, C., Controlling the Behavior of Knowledge Sources within SONIA," Proceedings of the 22nd Hawaii International Conference on System Sciences. 1989.

[21] Elleby, P., Fargher, H.E., and Addis, T.T., "Reactive Constraint-Based Job-Shop Scheduling," Technical Report, Department of Computer Science, Reading University, 1987.

[22] Conway, R.W., Maxwell W.L. and Miller, L.W., Theory of Scheduling, Addison-Wesley Publishing Company, 1967.

[23] Baker, K.R., Introduction to Sequencing and Scheduling, John Wiley & Sons, Inc. 1974.

[24] Halevi, Gideon, The Role of Computers in Manufacturing Processes, John Wiley & Sons, Inc. 1980

[25] Muth, J. and Thompson, G., Industrial Scheduling, Prentice-Hill Englewood Cliffs, NJ. USA, 1963.

[26] Ferguson, R.L and Jones, C.H., "A Computer Aided Decision System," Management Science 15(10), 1969, pp.550-561

[27] Wiest, J.D., "A Heuristic Model for Scheduling Large Projects with Limited Resource," Management Science 13(6). 1967, pp.359-377.[58] Liu, B., "A Reinforcement Approach to Scheduling," Proceeding of the 8th European Conference on Artificial Intelligence, 1988, pp.580-585.

[28] Fox, M.S., Constraint-Directed Search: A Case Study of Job Shop Scheduling, PhD Thesis, Carnegie-Mellon University, 1983.

[29] Waterman , D.A., A Guide to Expert Systems, Addison-Wesley, Reading MA. 1986.

[30] Kusiak, A. and Chen, M., "Expert Systems for Planning and Scheduling Manufacturing Systems," European Journal of Operational Research 34, 1988, pp.113-130.

[31] Fox, M.S. and Smith, S.F., "ISIS — a Knowledge-Based System for Factory Scheduling," Expert Systems 1(1), 1984, pp. 25-49.

[32] Ow, P.S and Smith, S.F., "Viewing Scheduling as an Opportunistic Problem-Solving Process," Carnegie-Mellon University, Working Paper, 1986.

[33] Smith, Stephen F., Fox, M.S and Ow, Peng, Si, "Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems," AI Magazine, Fall 1986. pp.46-61.

[34] Morton, T.E., et al., "MRR-STAR: PATRIARCH's Planning Module," Graduate School of Industrial Administration, Carnegie-Mellon University, Technical Report, 1986.

[35] Lepape, C., "SOJA: A Daily Workshop Scheduling System," Expert System 85, ed. Martin Merry, Cambridge University Press, 1985, pp.95-211.

[36] Bruno, B., Elia, A. and Laface, P., "A Rule-Based System to Schedule Production," IEEE Computer 7(19), 1986, pp.32-40.

[37] Erschler, J. and Esuirol, P., "Decision-Aid in Job Shop Scheduling: A Knowledge Based Approach," Proceeding of the 1986 IEEE International Conference on Robotic and Automation, San Francisco, 1986, pp.1651-1656.

[38] Subramanyam, S. and Askin, R.G., "An Expert System Approach to Scheduling in Flexible Manufacturing Systems," (1986), Flexible Manufacturing Systems: Methods and Studies, ed. A. Kusik, North-Holland, Amsterdam, 1986, pp.243-256.

[39] Kelley, J.E., "The Critical Path Method: Resources Planning and Scheduling," Chapter 21 in J. Muth and G. Thompson, Eds., Industrial Scheduling, Prentice-Hall, Englewood Cliffs, NJ., 1963.

[40] Tate, A., "Project Planning Using A Hierarchical Non-Linear Planner," D.A.I Research Report No. 25, University of Edinburgh, 1976.

[41] Stallman, R.M. and Sussman, G.J., "Forward Reasoning and Dependency Directed Backtracking," Artificial Intelligence 16, 1977, pp.135-196.

[42] Doyle, J., "A Truth Maintenance System," Artificial Intelligence 12, 1979, pp.231-272.

[43] Deis, P., Production and Inventory Management in the Technological Age, 1983.

[44] Liu, B., "Scheduling Via Reinforcement," International Journal for Artificial Intelligence in Engineering 3(2), 1988, pp.76-85.

[45] Blackstone, J.H., et al., "A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations," International Journal for Production Research 20(1), 1982, pp.27-45.

[46] Liu, Bing., "A Reinforcement Approach to Scheduling," Proceeding of the 8th Eurpean Conference on Artificial Intelligence, 1988, pp.580-585.

[47] Sacerdoti, E.D., "Planning in a Hierarchy of Abstaction Spaces," Advanced Papers of International Joint Conference on Artificial Intelligence, 1973, pp. 412-422, Palo Alto, CA, USA.

Bibliography

Allen, J., "Maintaining Knowledge about Temporal Intervals," CACM 26(11), 1983.

Allen, J., "Towards a General Model of Action and Time," Artificial Intelligence 23(2) 1984.

Ambros-Ingerson, J.A., "Relationships Between Planning and Execution," University of Essex, CSCM-25, 1986.

Barr, A. and Feigenbaum, E.A., The Handbook of Artificial Intelligence, Vol.1 William Kaufmann, 1981.

Barr, A. and Feigenbaum, E.A., The Handbook of Artificial Intelligence, Vol.2 William Kaufmann, 1982.

Barr, A., Bennett, J. and Clancey, W., "Transfer of Expertise: A Theme for AI Research," Rep.No. HPP-79-11, Heuristic Programming Project, Stanford University, 1979.

Ben-Arleh, D., "Knowledge Based Control System for Automated Production and Assembly," Modelling and Design of Flexible Manufacturing Systems, ed. Andrew Kusiak, Elsevier Science Publishers B. V., Amsterdam, 1986, pp.347-365.

Berlin, Daniel L.S., "SPAN: Integrating Problem Solving Tactics," Proceeding of Ninth International Joint Conference on Artificial Intelligence, 1985, pp.1047-1051.

Berry, W.L. and Rao, V., "Critical Ratio Scheduling: an Experimental Analysis," Management Science 22(2), 1975, pp.192-201.

Bobrow, D.G. and Winograd, T., "An Overview of KRL, a Knowledge Representation Language," Cognitive Science 1, 1977, pp.3-46

Bresina, J.L., Marsella, S.C. and Schmidt, C.F., "Predicting Subproblem Interactions," LCSR-TR-92, Laboratory for Computer Science Research, Rutgers University, 1987.

Buchanan, B.G., et al., "The Heuristic Refinement Method for Deriving Solution Structures of Proteins," Department of Computer Science, Stanford University, Report No, STAN-CS-86-115. 1986.

Charniak, E. and McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley Publishing Company, 1985.

Clancey, W. J., "Heuristic Classification," Department of Computer Science, Stanford University, Report No. STAN-CS-85-1066. 1985.

Cohen, P.R. and Feigenbaum, E.A., The Handbook of Artificial Intelligence 3, William Kaufmann, 1982.

Conway, R.W., "Priority Dispatching and Work-in-Process Inventory in a Job Shop," The Journal of Industry Engineering XVI(2), pp.123-130.

Conway, R.W., "Priority Dispatching and Job Lateness in a Job Shop," The Journal of Industry Engineering, XVI(4), pp.228-237.

Dannenbring, D.L., "An Evaluation of Flow Shop Sequencing Heuristics," Management Science 23(11), 1977, pp.1174-1182.

Dannenbring, D.L., "Procedures for Estimating Optimal Solution Values for Large Combinatorial Problems," Management Science 23(12), 1977, pp.1273-1283.

Davis, P.R., "Using and Re-using Partial Plans," University of Illinois at Urbana-Champaign, R-772: UILU-ENG-&&-2219, 1977.

Davis, R. and Lenat, D.B., Knowledge-Based System in Artificial Intelligence, McGraw-Hill, New York, 1982.

Dean, T. R., et al., "The Forbin Paper," Yale University, YALEU/CSD/RR #550, 1987.

Fahlman, S.E., "A Planning System for Robot Construction Tasks," Artificial Intelligence 5(1) 1974, pp.1-49.

Fike, R., Hart, P. and Nilsson, N., "Learning and Executing Generalized Robot Plans," Readings in Artificial Intelligence, ed. Nilson and Webber, Tioga Publishing, Palo Alto, CA, 1981, pp. 231-249.

Fox, M.S., "The Intelligent Management System: an Overview", The Robotics Institute, Carnegie-Mellon University, Technical Report. 1982.

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

Ginberg, M.L., "Decision Procedures," Department of Computer Science, Stanford University, Report No. STAN-CS-85-1064. 1985.

Goldstein, I.P. and Roberts, B., "Using Frames in Scheduling," In Artificial Intelligence: An MIT Perspective, ed. P.H Winston and R.H Brown, Cambridge, Mass., The MIT Press, 1979, pp.225-284.

Graves, S.C., "A Review of Production Scheduling," Operations Research 19(4), 1981.

Hayes, P.J., "A Representation for Robot Plans," Proceedings of Fourth International Joint Conference on Artificial Intelligence, 1975, pp.181-188, Tbilisi, USSR.

Hayes-Roth, F. and Lesser, V., "Focus of Attention in the Hearsay-II Speech Unserstanding System," Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, Aug 1977, pp.27-35.

Holloway, C.A and Nelson, R.T., "Alternative Formulation of the Job Shop Problem with Due Date," Management Science 20(1), 1973, pp.65-75.

Keng, N.P., Yun, D.Y.Y. and Rossi, M., "Interaction-Sensitive Planning System for Job-Shop Scheduling," Expert Systems and Intelligent Manufacturing (Proceedings of the Second International Conference on Expert Systems and the

Leading Edge in Production Planning and Control), ed. Michael D. Oliff. 1988, pp.57-69.

Jones, C.H., et al., "A Comparative Study of Management Decision Making from Computer-Terminal," AFIPS, Proceeding of the Spring Joint Computer Conference, 1970, pp. 599-605.

Lawrence, S.R., "An Experimental Investigation of Heuristic Scheduling Techniques," Graduate School of Industrial Adminstration, Carnegie-Mellon University, Technical Report, 1984.

Lenat, D.B., "AM: Discovery in Mathematics as Heuristic Search," in Knowledge-Based Systems in Artificail Intelligence, ed. Randall Davis and Douglas B. Lenat, MxGraw-Hill, New York, 1982, pp.1-225.

Liu, B., "Knowledge-Based Scheduling" DAI Discussion Paper No. 11, Department of Artificial Intelligence, University of Edinburgh, 1986.

Liu, B., "Constraint Decomposition in Scheduling," A forthcoming paper.

Liu, B., "Shop Floor Monitoring and Rescheduling," A forthcoming paper.

McCarthy, John and Hayes, Patrick J., "Some Philosophical Problems from the Standpoint of Artificial Intelligence," Machine Intelligence 4, ed. B Meltzer and D. Michie, Edinburgh University Press, Edinburgh, 1969.

McDermott, D.V., "Planning and Acting," Cognitive Science 2, 1978, pp.71-109

McDermott, D.V., "Planning Routes through Uncertain Territory," Artificial Intelligence 22, 1984, pp.107-156.

McDermott, D.V., "A Temporal Logic for Reasoning about Processes and Plans," Cognitive Science 6, 1982, pp.101-155.

Nicholson, T.A.J. and Pullen, R.D., "A Practical Control System for Production Schedules," International Journal of Production Research 9, 1971, pp.229-227.

Ow, P.S., Smith, S.F. and Rossi, M., "A Cooperative Scheduling System," Expert Systems and Intelligent Manufacturing (Proceedings of the Second International Conference on Expert Systems and the Leading Edge in Production Planning and Control), ed. Michael D. Oliff. 1988, pp. 43-56.

Raghavan, V., "An Expert System Framework for the Management fo Due-Dates in Flexible Manufacturing Systems," Expert Systems and Intelligent Manufacturing (Proceedings of the Second International Conference on Expert Systems and the Leading Edge in Production Planning and Control), ed. Michael D. Oliff. 1988, pp.235-247.

Ranky, P.G., "A Real-Time, Rule-Based FMS Operation Control Strategy in CIM Environment—Part I," International Journal of Computer Integrated Manufacturing 1(1), 1988, pp.55-72.

Rich, Elaine, Artificial Intelligence, McGraw-Hill, New York, 1983.

Rickel, J., "Issues in the Design of Scheduling Systems,"Expert Systems and Intelligent Manufacturing (Proceedings of the Second International Conference on Expert Systems and the Leading Edge in Production Planning and Control), ed. Michael D. Oliff. 1988, pp.70-89.

Rieger, C., "An Organization of Knowledge for Problem Solving and Language Comprehension," Artificial Intelligence 7, 1976, pp.89-127.

Rogers, P. and Williams, D.J., "Knowledge-Based Control in Manufacturing Automation," International Journal of Computer Integrated Manufacturing 1(1), 1988, pp.21-30.

Roenschein, S., "Plan Synthesis: A Logical Perspective," In Proceedings of International Joint Conference on Artificial Intelligence, 1981, pp.331-337

Shank, R.C. and Abelson, R.F., Scripts, Plans, Goals and Understanding, Lawrence Erlbaum Press, Hillsdale, New Jersey, USA, 1977.

Siklossy, L. and Dreussi, J., "An Effective Robot Planner that Generates Its Own Procedures," Advanced Papers of International Joint Conference on Artificial Intelligence, 1973, pp, 423-430, Palo Alto, CA, USA.

Smith, D.E., "Finding All of the Solutions to a Problem," Proceeding of AAAI3, 1983, pp.373-377.

Smith, H.T. and Crabtree, R.G., "Interactive Planning: a Study of Computer Aiding in the Execution of a Simulated Scheduling Task," International Journal of Man-Machines Studies 7, 1975, pp.213-231,

Smith, S.F., "A Constraint-Based Framework for Reactive Management of Factory Schedules," Proceedings International Conference on Expert Systems and the Leading Edge in Production PLanning and Control, 1987.

Smith, S.F. and Hynynen, J.E., "Integrated Decentralization of Production Management: an Approach for Factory Scheduling," Proceedings 1987 Symposium on Integrated and Intelliigent Manufacturing, ASME Annual Winter Conference, 1987.

Smith, S.F and Ow, Peng, SI, "The Use of Multiple Problem Decomposition in Time Constrained Planning Tasks," Proceedings of Ninth International Joint Conference on Artificial Intelligence, 1985, pp.1013-1015.

Smither, T., "Artificial Intelligence and Product Creation: An AI-Based View," DAI Research Paper No. 342. Department of Artificial Intelligence, University of Edinburgh, 1987.

Sussman, G.A., A Computational Model of Skill Acquisition, American Elsevirer, New York, 1975.

Tate, A., "Interacting Goals and Their Use," Proceedings of Fourth International Joint Conference on Artificial Intelligence, 1975, pp. 215-218. Tbilisi, USSR.

Tate, A.,"Goal Structure Capturing the Intent of Plans," Proceedings of the Six European Conference on Artificial Intelligence, pp. 273-276, Pisa, Italy, 1984.

Tate, A., "A Review of Knowledge-Based Planning Techniques," Expert Systems 85, ed. Martin Merry, Cambridge University Press, 1985, pp.89-111.

Ward, B. and McCalla, G.,"Error Detection and Recovery in A Dynamic Planning Envirnment," Proceedings of the National Conference on Artificial Intelligence, AAAI, Pittsburgh, Pennsylvania, August 1982, pp.172-175

Wilensky, R., Planning and Understanding, Addison-Wesley Publishing Company, 1983, pp. 29-40.

Wilensky, R., LISPcraft, Norton, New York, 1984.

Winston, P.H., Artificial Intelligence, Second Edtion, Addison-Wesley Publishing Company, 1984.

Zaloom, V., "On the Resource Constrained Project Scheduling Problem," AIIE Tranaction III(4), 1971, pp.303-305.

Appendix

Papers Published

Scheduling Via Reinforcement. *International Journal for Artificial Intelligence in Engineering 3(2), pp.76-85, 1988*

A Reinforcement Approach to Scheduling. *Proceeding of the 8th European Conference on Artificial Intelligence, pp.580-585, 1988.*