APPROXIMATE MODELS OF JOB SHOPS

Alexandros Diamantidis B Sc

M Sc in Computer Applications

Dublin City University

Supervisor Dr Micheal O'hEigeartaigh

August 1999

August 1999

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of M.Sc. is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed:____________________ Date: 7/8/1999.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOLEDGEMENTS

I would like to extend my heartfelt gratitude to my supervisor Dr Michcal O'hEigcartaigh for all his help and guidance throughout the period of my research Without his assistance I would not have been able to reach this stage

My sincere gratitude to Prof E O'Kelly for his invaluable suggestions and continuous support I would like also to thank Assistant Prof Chr Papadopoulos for his invaluable suggestions and quidance, which started six years ago when I was a student at Department of Mathematics in University of Aegean and continued until now

I would like to thank also Prof Michael Ryan and the School of Computer Applications in general for providing me with the opportunity and the necessary funding to undertake this research

I would like also to thank my colleague Theodore Stamoulakatos for his suggestions on C++ coding Finally a great thank deserve to all the postgraduate students of the School of Computer Applications for making my research life exciting

Abstract

Scheduling can be described as "the allocation of scarce resources over time to perform a collection of tasks" They arise in many practical applications in manufacturing, marketing, service industries and within the operating systems of computers

Scheduling problems are frequently encountered in various activities of every day life

They exist whenever there is a choice of the order in which a number of tasks can be performed Some examples are scheduling of classes in academic institutions, jobs in manufacturing plants, patients on test facilities in health institutions and programs to be run at a computing centre The desire to perform the tasks in a special order to achieve some objective is what makes scheduling problems important

In this thesis we will use the machine shop terminology, even though the actual situations that give rise to scheduling problems are wide and varied

Since a complete description of a real machine shop would be too detailed to serve as a conceptual basis for any meaningful analysis, we will adopt a simplified model consisting of a job shop and a despatch area through which jobs are received from outside and then passed to the job shop

Such a model can adequately reflect the aspects of real machine shops that are important for predicting performance

The performance of such a job shop system models is normally measured by either the production capacity or mean tardiness or the mean number in the system, in the job shop and in the despatch area

Scheduling problems differ in

- input, the manner in which the jobs arrive at the system
- despatch policy, the policy by which the jobs are despatched to the shop, and
- routing, the order in which the jobs go from one machine centre to the other in the job shop

CHAPTER 1

INTRODUCTION

1 0 A brief history of scheduling theory

Scheduling theory is concerned with the practical problem of allocating (scarce) resources over time to perform a collection of tasks, with a view to minimising an evaluation function [B74]

This rather general definition of the term does convey two different meanings that are important to understand the necessity of scheduling in our lives First, scheduling is a decision-making function In this sense the process of determining a schedule and much of what we learn about scheduling can apply to other kinds of decision making and therefore has general practical value
Second, scheduling is a body of theory it is a collection of principles, models, techniques, and logical conclusions that provide insight into the scheduling function In this sense, much of what we learn about scheduling can apply to other theories and therefore has general conceptual value

The problem being investigated is normally cast in terms of a mathematical model Seminal work [K76] in developing a categorisation of scheduling problems has enabled researchers in combinatorial optimisation co-ordinate their efforts in the design of good algorithms A large range of problems of practical interest has been wholly or partly solved to date

However, as new problem classes are identified, there is a necessity to develop new models and solution techniques on a continual basis

The theoretical perspective is predominantly a quantitative approach, one that attempts to capture problem structure in concise mathematical form In particular, this quantitative approach begins with a translation of decision-making goals into an explicit objective function and decision-making restrictions into constraints Ideally, the objective function should consist of all costs in the system that depends on scheduling decisions In practice, however, such costs are often difficult to measure, or even to identify completely

The most important elements in scheduling models are resources and tasks Tasks compete for resources A task is described by its resource requirement, its duration, the time at which it may be started and the time at which it is due to be completed

Because many of the early developments in the field of scheduling were motivated by problems arising in manufacturing, the vocabulary of manufacturing is still employed when describing scheduling problems Thus resources are usually called "machines" and basic task modules are called "jobs" Jobs may consist of several elementary tasks that are interrelated by precedence restrictions, such elementary tasks are referred to as "operations"

1.1 Importance of scheduling problems

Scheduling problems are encountered in various activities of everyday life They exist whenever there is a choice of the order in which a number of tasks can be performed Some examples are scheduling of classes in academic institutions, jobs in manufacturing plants, patients on test facilities in health institutions and programs to be run at a computer centre The desire to perform the tasks in a special order to achieve some objective is what makes scheduling problems important [S79]

Scheduling problems are also important because the scheduling field has become a focal point for the development, application and evaluation of combinatorial procedures, simulation techniques, network methods and heuristic solution approaches

The selection of an appropriate technique depends on the complexity of the problem, the nature of the model and the choice of the criterion, as well as other factors, in many cases it is appropriate to consider several alternative techniques For this reason, scheduling theory is perhaps as much the study of methodologies as it is the study of models

Because scheduling is a body of a theory (a collection of principles, models, techniques, and logical conclusions) much of what we learn about scheduling can apply to other theories and therefore has general conceptual value

1 2 Outlıne of the thesıs

The thesıs ıs made up of seven chapters The second chapter the lıterature revıew ıs presented along wıth the classıficatıon of schedulıng problems

In chapter 3 the descrıptıon of our model ıs presented as well as prevıous studıes on relatıve models In chapter 4 the Mıxed Integer Programmıng (MIP) formulatıon ıs descrıbed, whereas ın chapter 5 we descrıbe the algorıthm that we developed for schedulıng groups of jobs on a sıngle machıne (JGA)

In chapter 6 we descrıbe an algorıthm that ıs used for schedulıng a set of jobs on a sıngle machıne, whereas ın chapter 7 we evaluate the performance of both algorıthms and we make some suggestıons for further research

CHAPTER 2

LITERATURE REVIEW ON SCHEDULING THEORY

Review of scheduling theory

Scheduling can be described as "the allocation of scarce resources over time to perform a collection of tasks" They arise in many practical applications in manufacturing, marketing, service industries and within the operating systems of computers

Scheduling tasks are characterised by the

- Environment in which they are defined (e g single or multiple machine context)
- Job characteristics (e g presence of deadlines, release dates)
- Optimality criteria (e g Cmax, Lmax)

In this paper we present a literature review of recent advances in scheduling theory

2 0 Introduction

Scheduling has been described as "the allocation of resources over time to perform a collection of tasks"([B74], p2) As the definition implies, scheduling theory arises within the realm of Combinatorial Optimisation (CO) and is closely related to partitioning and packing problems

Scheduling theory is concerned primarily with mathematical models that relate to the scheduling function This area has been researched very heavily since 1950 and many excellent review articles chart progress within the domain over that time The research direction has been driven by practical applications and scheduling problems are classified by the

- Environment in which they are defined (e g single or multiple machine context)
- Job characteristics (e g presence of deadlines, release dates)
- Optimality criterion, which is to be minimised (e g Cmax, Lmax)

Ideally, the optimality criterion should consist of all costs in the system that depends on the scheduling decisions In practice, however such costs are often difficult to measure, or even to identify completely According to [B74] three types of decision-making goals are prevalent in scheduling

- Efficient utilisation of resources
- Rapid response to demands
- Close conformance to prescribed deadlines

By virtue of the classification scheme used to describe them, scheduling problems are easy to describe However, they include many NP-hard problems and the field has become a focal point for the development, application, and evaluation of combinatorial procedures, simulation techniques, network methods, and heuristic solution approaches

2 1 Job shop models

Since a complete description of a real machine shop would be too detailed to serve as a conceptual basis for any meaningful analysis, we will adopt a simplified model consisting of a Job Shop and a dispatch area Jobs are received through dispatch area from outside and then passed to the job shop

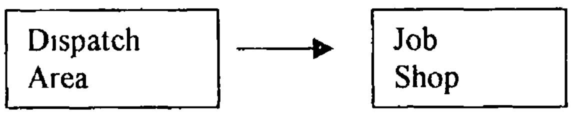

Figure 1 Job shop system

Such a model can adequately reflect the aspects of real machine shops that are important for predicting performance The performances of such Job Shop system models is normally measured by either the production capacity or mean tardiness or the mean number of jobs in the system in the Job Shop and in the despatch area However occasionally other system measures, which will be discussed later, are also used [S79]

Problem Variables

- N The number of jobs to be scheduled
- M The number of machines (each job is assumed to visit each machine once)
- d_i Deadline for job i (job i must be completed by date d_i)
- d_i Due date of job i (it is desirable that job i be completed before the date d_i)
- r_i Release date for job i (job i can not be started before date r_i)
- p_{ij} Setup and processing time of job i on machine j

Pre-emption (pmtn) is the ability to start or stop the processing of a job arbitrarily often It is a watershed in scheduling problems if it is allowed, it tends to make scheduling easy It is a characteristic of computer related problems, such as the scheduling of tasks within an operating system, it is rarely present in workshop problems

Solution-Dependent Measures

- C_i Time at which job i is completed
- F_i The length of time job i is in the shop (flow time)
- L_i Lateness (C_i-d_i)
- T_i Tardiness (max{0,L_i}, i e positive lateness values)

In general we assume that all jobs are in the shop and ready for processing at time 0, and hence flow time and completion time are the same

Description of a Shop

In a shop-scheduling problem we are given a set of jobs $J=\{J_1,J_2, \quad ,J_n\}$ a set of machines $M=\{M_1,M_2, \quad ,M_m\}$ and a set of operations $O=\{O_1, \quad ,O_t\}$ each operation $O_k \in O$ belongs to a specific job $J_j \in J$ and must be processed on a specific machine $M_i \in M$ for a given amount of time p_k, which is a non-negative integer At any time, at most one operation can be processed on each machine, and at most one operation of each job can be processed [K76]

According to [S79], scheduling problems differ in

- Routing - the order prescribed for jobs on the machines in the shop
- Input - the manner in which the jobs arrive at the system
- Dispatch policy - the manner in which the jobs are dispatched to the shop

2 2 Routing

A shop could be characterised by the following broad divisions

- Open Shop - jobs can be processed on the machines in any order
- Job Shop - individual jobs have a prespecified machine sequence
- Flow Shop - all jobs follow the same prespecified machine sequence

2 3 Input

In [S79], scheduling problems are classified as static and dynamic, depending on the job arrival pattern In a **Static Job Shop,** a certain number of jobs arrive simultaneously to a system that is idle and is immediately available for work No additional jobs will be assigned to the system until they are dispatched to the shop This prescheduling of jobs before dispatching may be carried out taking into account the storage capacity of the Job Shop, the processing times of the operations , due dates and so on

This preschedule stage can be used to obtain a dispatch schedule and assign priorities for each job on each machine If no conflict arises in the shop with respect to the priorities and dispatch schedules, the whole operation can be carried out according to the preschedule However if conflicts arise due, inter alia, to machine breakdowns, server vacations or uncertain processing times, it may become necessary to practise **shop level scheduling** (that is, priority assignments are made by the machine operator or shop floor supervisor)

The shop level scheduling can be classified into two categories local and global **Local scheduling** rules assign priorities to jobs at a machine based on the immediate status of the jobs at that machine, **global scheduling** rules require information about the status of some aspects of the system beyond the local boundaries of that machine

In a **Dynamic Job Shop** system, jobs arrive intermittently at times that are predictable only in a statistical sense The jobs may belong to one or more classes

2 4 Dispatch policies

In a **Pseudo-Static Job Shop,** the dynamic scheduling problem is converted into a sequence of static problems At review times all jobs in the Job Shop and dispatch area are prescheduled using static rules All these jobs are treated as a new batch, in the same way as those in a static scheduling problem Any job entering the system after a review time must wait in the dispatch area until the next review time No shop level scheduling is permitted unless it is required to resolve conflicts due to prescheduling priorities

In a **Pure Dynamic Job Shop**, each job on arrival to the system enters the shop immediately and only shop level scheduling is permitted When jobs in a pure dynamic Job Shop are processed in the order of their arrival to the machines, the system is typically treated as the classical Jackson type queuing network model In order to improve the performance of the Job Shop, jobs may be scheduled at each machine according to some priority rules such as shortest processing time (SPT)

Pseudo-Dynamic Job Shop models represent systems where jobs can be held at the dispatch area and control exercised at the prescheduling and shop levels, depending on the type of information available

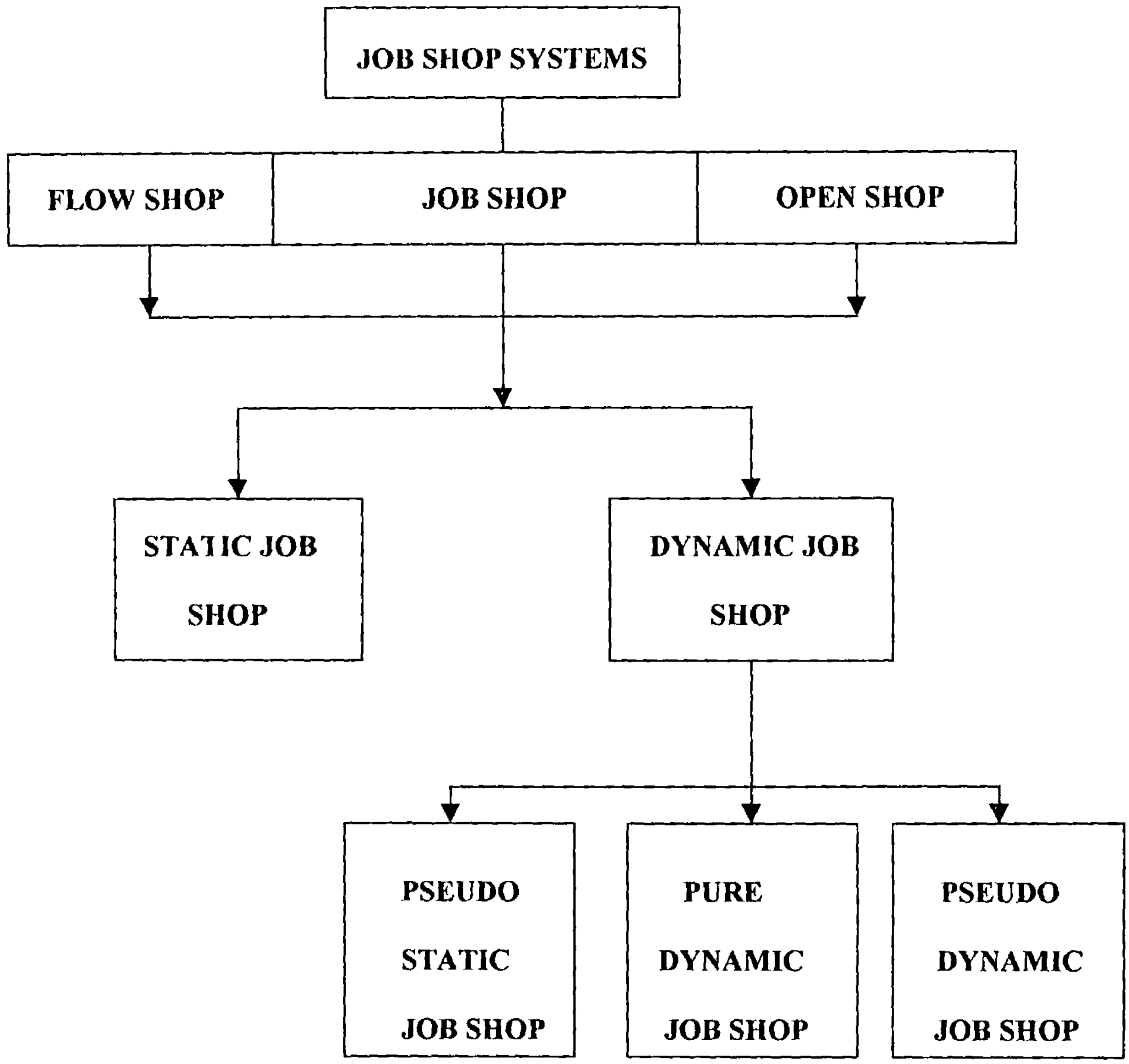

Figure 2 Classification of job shops and scheduling systems

2 5 General assumptions

Following [S79], [K76] and [AS93], the following assumptions will be made

Job Based Assumptions

- The set of jobs J is known and fixed
- Jobs arriving in the system go directly to the dispatcher and each job is released to the shop as soon as it enters the dispatch area
- All jobs are available at the same instant and independent
- Each job consists of specified operations, each of which is performed by only one machine at a time
- Each job requires a finite process time for each operation The processing times of all jobs at a machine are identically and independently distributed
- Each job can be in each one of three states
    - Waiting for the next machine
    - Being operated by a machine
    - Having passed its last machine
- Each job is processed by all the machines assigned to it
- All jobs are equally important
- All jobs remain available during an unlimited period

Machine Based Assumptions

- The set of machines M is known and fixed
- Each machine is continuously available for processing jobs and there are no interruptions due to breakdowns, maintenance or other such cases
- All machines remain available during an unlimited period
- Each machine in the shop operates independently of the other machines and thus is capable of operating at its own maximum output rate
- Each machine can be in each one of three states
    - Waiting for the next job
    - Operating on a job
    - Having finished its last job
- All machines are equally important
- Each machine processes all the jobs assigned to it
- Each machine processes one job at a time

Operating Policies

- Each job is considered as an indivisible entity even though it may be composed of a number of individual units
- Each operation once started must be completed without interruption (If preemption is allowed, this assumption will be altered)
- All processing times are fixed and sequence-independent
- The processing order per job is known and fixed
- Each job once accepted, is processed to completion, without cancellation

- Each machine is fully allocated to the jobs under consideration

Scheduling Policies

- **SPT (Shortest Processing Time)** Select a job with minimum processing time
- **EDD (Earliest Due Date).** Select a job due first
- **FCFS (First Come, First Served).** Select a job that has been in the workstation's queue the longest
- **FISFS (First In System, First Served).** Select a job that has been on the shop floor the longest
- **S/RO (Slack per Remaining Operation).** Select a job with the smallest ratio of slack to operations remaining to be performed
- **Covert** Order jobs based on ratio of slack-based priority to processing time
- **LTWK (Least Total Work)** Select a job with smallest total processing time
- **LWKR (Least Work Remaining)** Select a job with smallest total processing time for unfinished operations
- **MOPNR (Most Operations Remaining)** Select a job with the most operations remaining in its processing sequence
- **MWKR (Most Work Remaining).** Select a job with the most total processing time remaining
- **RANDOM (Random)** Select a job at random

- **WINQ (Work In Next Queue)** Select a job whose subsequent machine currently has the shortest queue
- **SPTT (Truncated Shortest Processing Time).** In SPTT scheduling discipline, jobs are divided by the controller into two classes such that jobs with processing time less than or equal to α belong to class 1and the rest to class 2 Here α is the boundary point Higher priority is given to class 1 However within class 1 jobs are selected according to SPT and within class 2 according to FCFS
- **{2C-NP} (Two Class Non-Preemptive Priority).** In 2C-NP, jobs are divided by the controller into two classes as in SPTT However within each class jobs are selected according to FCFS
- **2L-SPT (Two Level Shortest Processing Time)** In 2L-SPT, jobs are divided by the controller into two classes A job is randomly assigned to class 1 with probability f and to class 2 with probability 1-f Class 1 jobs are given higher priority and within each class SPT discipline is used

2.6 Problem classification

In [DLR81], scheduling problems are classified using three characteristics $\alpha|\beta|\gamma$, where α is the machine environment, β defines the job characteristics and γ is the optimality criterion that is to be minimised

Machine Environment

We describe here the first field $\alpha = \alpha_1\alpha_2$ which specifies the machine environment

Let o denote the empty symbol If $\alpha_1 \in \{o, P, Q, R\}$, each job J_j consists of a single operation that can be processed on any machine M_i, the processing time of J_j on M_i being p_{ij}

There are four cases to consider

- $\alpha_1 = o$ Single machine, $p_{1j} = p_j$
- $\alpha_1 = P$ Identical parallel machines, $p_{ij} = p_j$ ($i = 1, \quad ,m$)
- $\alpha_1 = Q$ Uniform parallel machines, $p_{ij} = p_j / q_i$ for a given speed q_i of M_i ($i = 1, \quad ,m$)
- $\alpha_1 = R$ Unrelated parallel machines

If $\alpha_1 = O$ we have an open shop, in which each J_j consists of a set of operations $\{O_{1j}, \quad , O_{mj}\}$ O_{ij} has to be processed on M_i during p_{ij} time units However, the order in which the operations are executed is immaterial

If $\alpha_1 \in \{F, J\}$, an ordering is imposed on he set of operations corresponding to each job If $\alpha_1 = F$, we have a Flow Shop and if $\alpha_1 = J$, we have a Job Shop If α_2 is a positive integer, then m is a constant and equal to α_2 If $\alpha_2 = o$ then m is assumed to be variable

Job Characteristics

The second field $\beta \in \{\beta_1, \quad ,\beta_5\}$ defines the job characteristics

- $\beta_1 \in \{pmtn, o\}$

$\beta_1 =$ pmtn Preemption (job splitting) is allowed the processing of any operation may be interrupted and resumed at a later time

$\beta_1 = o$ No preemption is allowed

- $\beta_2 \in \{prec, tree, o\}$

$\beta_2 =$ prec A precedence relation $\rightarrow$ between the jobs is specified $J_j \rightarrow J_k$ requires that J_j be completed before J_k can start

$\beta_2 =$ tree G is a rooted tree with outdegree at most one for each vertex

$\beta_2 = o$ No precedence relation is specified

- $\beta_3 \in \{r_j, o\}$

$\beta_3 = r_j$ Release dates that may differ per job are specified

$\beta_3 = o$ All $r_j = 0$

- $\beta_4 \in \{m_j < m, o\}$

$\beta_4 = m_j < m$ A constant upper bound on m_j is specified (only if $\alpha_1 = J$)

$\beta_4 = o$ All m_j are arbitrary integers (Where $\{O_{1j}, \quad ,O_{mj}\}$ is a set of operations that each J_j is consisted of) O_{ij} has to be processed on M_i during p_{ij} time units

- $\beta_5 \in \{p_{ij} = 1, o)$

$\beta_5 = p_{ij} = 1$ Each operation has unit processing time

(if $\alpha_1 \in \{o, P, Q\}$, we write $P_j=1$ and if $\alpha_1=R$, $p_{ij}=1$ will not occur)

$\beta_5 = o$ All p_{ij} (p_j) are arbitrary integers

Optimality Criteria

The third field $\gamma \in \{f_{max}, \Sigma f_j)$ refers to the optimality criterion which is to be minimised The optimality criteria most commonly chosen in the literature are

- $f_{max} \in \{C_{max}, L_{max}\}$,

  where $f_{max} = \max_j (f_j(C_j))$ with $f_j(C_j) = C_j, L_j$ respectively

- $\Sigma f_j \in \{\Sigma C_j, \Sigma T_j, \Sigma U_j, \Sigma w_j C_j, \Sigma w_j T_j, \Sigma w_j U_j\}$,

  where $\Sigma f_j = \Sigma f_j(C_j)$ with $f_j(C_j) = C_j, T_j, U_j, w_j C_j, w_j T_j, w_j U_j$, respectively

  (all these factors will be defined in the next section)

For example $R|pmtn|\Sigma C_j$ Minimise total completion time on a variable number of unrelated parallel machines, allowing preemption The complexity of this problem is unknown

Other objectives are minimising

- Average flow time, $F = (F_1+F_2+ \quad +F_N) / N$ (N = total number of jobs)
- Time required to complete all jobs (C_{max}, also referred as makespan)
- Average tardiness, $T = (T_1+T_2+ \quad +T_N) / N$
- Maximum tardiness (T_{max})
- Number of tardy jobs, $U_1+U_2+ \quad +U_N$, where U_i is 1 if $T_i>0$ and 0 otherwise
- Weighted sum of job completion times, $w_1C_1+w_2C_2+ \quad +w_NC_N$

  where each job has a specified weight w_i
- Total tardiness, $T_1+T_2+ \quad +T_N$
- Sum of Cost Functions, $f_1(C_1)+f_2(C_2)+ \quad +f_N(C_N)$, where for each job j there is specified a cost function f_j

2 7 Studies of dynamic job shop systems and related models

Most methods proposed for solving the job shop scheduling problem are of an enumerative type, and use a disjunctive graph formulation proposed by [RS64] Nevertheless, other approaches have been tested most of them based on an active schedule generation or mixed integer programming (MIP) formulation

In this section we will expose some ideas, of some researchers about the job shop scheduling problem These ideas are taken from articles in magazines that have been published the last four years

2 7 1 Disjunctive graph formulation

The model can be modelled by a disjunctive graph K=(G,D), where G=(X,U) is a conjunctive graph associated with the job sequences Most methods proposed for solving the job shop-scheduling problem are of an enumerative type, and use a disjunctive graph formulation proposed by [RS64] Nevertheless, other approaches have been tested most of them based on an active schedule generation or mixed integer programming (MIP) formulation

- X is the set of vertices which represent the tasks to be performed, including the fictitious *start* and *finish* tasks,
- U is the set of conjunctive arcs representing the order in which the tasks belonging to the same jobs should be performed
- D is the set of disjunctive arcs, and more precisely the set of pairs of opposite directed lines (i e arcs) which represent the possible precedence constraints among tasks belonging to different jobs but performed on the same machine

Two operations i and j, executed by the same machine, can not be simultaneously processed So we associate with them a pair of disjunctive arcs or disjunction $[i,j]=\{(i,j),(j,i)\}$

Usually o and * denote two dummy operations associated with the beginning and the end of the schedule In the following p_i denotes the processing time of operation i A schedule on a disjunctive graph K= (G,D) is a set of starting times $T=\{ t_i \; i \in X \}$ such that

- The conjunctive constraints are satisfied

$$t_j - t_i \geq p_i \quad \forall (i,j) \in U$$

- The disjunctive constraints are satisfied

$$t_j - t_i \geq p_i \text{ or } t_i - t_j \geq p_j \quad \forall (i,j) \in D$$

To built a schedule, we have to replace each disjunctive arc $[i,j]$ by either (i,j) or (j,i), and thus to choose an operating sequence for each machine

<u>2.7 2 Mixed integer programming formulation</u>

A large number of MIP formulations have been proposed by a number of authors [F82] A new MIP formulation that has been recently used by [AC91] is presented here Keeping the notation defined above, the problem can be formulated as follows

Minimize C_{max}

Subject to

$$\forall i \in X, \; t_i \geq 0,$$

$$\forall i \in X, \; C_{max} \geq t_i + p_i$$

$$\forall (i,j) \in U, \; t_j \geq t_i + p_i$$

$$\forall [i,j] \in D, \; t_j \geq t_i + p_i \text{ or } t_i \geq t_j + p_j$$

This disjunctive programming problem leads to the following MIP formulation by introducing a binary variable y_{ij} and setting the new constraints

$$\forall\ [i,j] \in D,\ t_i \geq t_j + p_j - K y_{ij},\ t_j \geq t_i + p_i - K(1 - y_{ij})$$

$$\forall\ [i,j] \in D,\ y_{ij} \in \{0,1\},$$

where K is some large constant, and $y_{ij}=1$ if and only if i is scheduled before j, and 0 otherwise

2.7.3 Job grouping

Economies of scale are fundamental to manufacturing systems With respect to scheduling this phenomenon manifests itself in efficiencies gained from grouping similar jobs together Job grouping [WB95], [AW97] are techniques that have been tested on the job shop scheduling problem In both cases jobs are grouped into families where jobs in the same family share a setup (a job does not need a setup when following another job from the same family) but a known "family setup time" is required when a job follows a member of some other family

In [WB95] an overview of research results for scheduling groups of jobs on a single machine is presented These results fell into three categories, according to the scheduling model

- Family scheduling with item availability
- Family scheduling with batch availability
- Batch processing

In the first model a job becomes available for delivery to the next stage as soon as it completes processing A simplifying assumption for family scheduling is that precisely f setups in the schedule are needed, one for each family (f is the number of families) This assumption is called GT assumption

The authors show that the F_w problem and the L_{max} problem are easy to solve when the GT assumption holds, otherwise, the F_w is open and the L_{max} problem is known to be NP-hard One useful direction for further research would be to resolve the complexity of the F_w problem If it is NP-hard, then another researchable area would be the development of algorithms for either problem Some sufficient conditions for the optimality of the GT solution are also presented

Next they reviewed the major results for the family scheduling model with batch availability, which characterize the solution of the F problem when there is one family They applied the same principles to develop a solution to the L_{max}

problem when there is one family The generalization to multiple families is a challenging area for future work, as is the one-family problem with the F_w objective

They also highlighted several results for the batch processing model which has received attention only recently in the scheduling literature They focused on models involving dynamic job arrivals, in light of the fact that the static version of the batch processing problem is often trivial Two broad areas for future work appear fertile One involves relaxing the assumption of a single machine and the other area involves criteria other F_w and L_{max}

2 7 4 Branch and Bound methods

Branch and bound techniques have been tested on the job shop scheduling problem [AW97] analyses a model of a single machine scheduling problem with family setup times, arbitrary earliness and tardiness job penalty rates, and an unrestricted common due date is analysed to minimize total weighted earliness and tardiness cost These rates are assessed on a per-period basis when the completion time deviates from its due date

The interesting point of this work is that it combines the features of family setup times (job grouping) with earliness / tardiness cost They have generalized properties from the literature [HP91] that help characterize the form of optimal schedules and they have defined an efficient method for calculating a lower bound on the optimum The properties and lower bounding methods are incorporated into a branch and bound and a beam search procedure

Each node in the tree (with the exception of the bottom-level) corresponds to a partial schedule When an unsequenced job is added to a partial schedule S, it is added to either the beginning of E or the end of T (where E and T are the ordered set of jobs that complete no later than time d and after time d respectively)

The branch and bound algorithm employs a depth-first strategy A node in r^{th} level of the branch and bound tree corresponds to a partial sequence with r jobs For each node at level r, there are two nodes emanating for each unsequenced job one for the first available early position and one for the first available tardy position The nodes that can not be fathomed by some dominance conditions are listed in nondecreasing order of lower bounds The node at the top of the list is selected for branching

Beam search is a heuristic branch and bound procedure that does not necessarily evaluate the complete branch and bound tree Thus, the approach sacrifices a guarantee of optimality for gains in speed and reduced memory requirements At each level only a limited number of nodes are selected for branching, the rest are permanently discarded The number of nodes selected for branching is called the beam width

2 7 5 Simulated Annealing

Simulated Annealing [LAL92] is one of the most important local search techniques that have been tested on the job shop scheduling problem In [LAL92] an approximation algorithm is presented for the problem of finding the minimum makespan in a job shop The algorithm is based on simulated annealing, a generalization of the well known iterative improvement approach to combinatorial optimization problems and is a more general approach based on the easily implementable simulated annealing algorithm [KGV83]

The innovation of the algorithm involves the acceptance of cost-increasing transitions with a nonzero probability to avoid getting stuck in local minima That probabilistic element of the algorithm makes simulated annealing a significantly better approach than the classical iterative improvement method on which is based The neighborhood structure is based on critical path rearrangement

A transition is generated by reversing the sequencing order of two critical operations

[LAL92] establishes the asymptotic convergence in probability to a global minimal solution of a simulated annealing procedure using the first neighborhood mentioned above In comparison with other heuristic methods, simulated annealing yields consistently good solutions Simulated annealing has the disadvantage of large running times which can be compensated for by the simplicity of the algorithm, by its ease of implementation, by the fact that it requires no deep insight into the combinatorial structure of the problem, and, of course, by the high quality of solutions it returns

2 7 6 Tabu Search techniques

Other local search techniques that have been tested on the job shop scheduling problem are Tabu Search techniques [DT93] In [FS] a new heuristic method based on the Tabu Search technique for solving the n-job m-machine job shop scheduling problem to minimize the makespan is presented

The authors start from an initial solution by sequencing randomly the jobs to the machines Given a sequence s, they define N(s) as being the set of all feasible sequences which can be obtained from s by applying a method which firstly constructs a priority list of jobs, secondly selects the job on the first position of the priority list and then assigns this job to the machine on the first position of the job's operations sequence

After that a job on the second position is selected and assigned to the machine on the first position of the job's operations sequence, and so on Because the objective function is the makespan, the best neighbour is selected as the sequence that minimizes the makespan all over sequences in N(s) and which does not lead to tabu moves

The algorithm is sometimes simplified by examining neighbours and taking the first one that improves the current solution If there is no move that improves the solution (or if all improving solutions are tabu) then the whole set of neighbours is examined If all the generared neighbors do not improve the solution or all the improving neighbors are tabu, all neighbours are examined

The procedure is stopped when Nmax iterations have been performed without improving the current solution (where Nmax is a parameter of the algorithm and can be set by experimentation) It was observed that the better the initial solution, the better the results and also the smaller the number of iterations Thus a an idea for future work may be to find better ways of generating a neighbour, testing for the best parameter settings, and finding a better starting solution In comparison with other heuristic algorithms, tabu search yields quite good solutions and is less time-consuming than simulated annealing

2 7 7 Truncated Branch and Bound methods

One of the most efficient approximate methods proposed so far is probably the Shifting Bottleneck Procedure presented in [ABZ88] Starting with the initial job shop scheduling problem, the authors optimally sequence one by one the machines, using Carlier's (1982) [C82] algorithm for the one machine problem At each optimization step, heads and tails adjustments are computed The order in which the machines are sequenced depends on a bottleneck measure associated with them Each time a new machine is sequenced, they attempt to improve the operating sequence of all previously scheduled machines in a reoptimization step This procedure is embedded in a second heuristic of an enumerative type, for which each node of the search tree corresponds to a subset of sequenced machines

CHAPTER 3

MODEL DESCRIPTION

3.0 Description of our model

We consider a model that is based on single-machine scheduling models that incorporate benefits from job grouping

In some settings, the grouping of jobs is a desirable or necessary tactic because of some technological feature of the processing capability The motivation for grouping sometimes relates to the existence of changeover times, or set-up times on the machine

Suppose that jobs each belong to a particular family, where jobs in a family tend to be similar in some way, such as their required tooling or their container size As a result of this similarity, a job does not need a set-up when following another job from the same family, but a known "family set-up time" is required when a job follows a member of some other family This is called **family scheduling model**

In the family scheduling model, a machine is assumed capable of processing at most one job at a time We use the pair (i,j) to refer to job j of family i We let f denote the number of families, n the number of jobs, and n_i the number of jobs belonging to family i

In addition $p_{i,j}$ and $w_{i,j}$ denotes the processing time and weight of job (i,j) Thus $n_1 + n_2 + \quad + n_f = n$ In addition, s_i denotes the setup time required to process a job in family i following a job in some other family In principle any family scheduling model can be viewed as a single-machine model with sequence dependent setup times If a job follows a member of the same family, then its setup time is zero otherwise its setup time is s_i, the family setup time

We know that sequence-dependent set-up times tend to make solutions difficult to find However, by exploiting the special structure of family scheduling, we can sometimes avoid the enumerative techniques that would ordinarily be required A simplifying assumption for family scheduling is the requirement of precisely f set-ups in the schedule, one for each family Such a requirement may reflect the fact that the set-ups are much longer than the job processing times, or it may result from a desire to minimize the time spent on set-up in situations where capacity is scarce It may also be imposed simply to make the problem more tractable We refer to this assumption as the **GT** assumption

Each family is treated as a single entity, or composite job with processing time

$$p_i = \sum_{j=1}^{n} p_{i,j} \text{ and weight } w_i = \sum_{j=1}^{n} w_{i,j}$$

We consider the problem of assigning due-dates and sequencing a given set of jobs on a single machine There will be penalties for completing jobs either ahead or behind their scheduled dates The objective is to minimize a function of missed due dates

We are concerned with the optimal sequencing of a set of jobs to minimise a penalty of deviation from the desired due-dates It is coupled with the optimal assignment of due-dates to the set of jobs to be processed by a single machine Given a set of families of jobs with deterministic processing times and the same ready times, the problem is to find the optimal common flow allowance k^* and the optimal job sequence σ^* to minimize a penalty function of missed due dates
It is assumed that penalty will not occur if the deviation of job completion from the due-date is sufficienlty small

Scheduling against due-dates has been a popular research topic in the scheduling literature for many years [BS90], [BGG88], [B87], [HP89] It attracts the attention of both Operational Research researchers and practitioners for two reasons The combinatorial nature of the due-date scheduling problem poses a great theoretical challenge to researchers who are trying to develop time-efficient algorithms to solve the problem in an elegant manner

The results of due-date scheduling research have significant practical value in the real world It is evident that the failure of completing a job on its promised delivery date gives rise to various penalty costs Completing a job early means having to bear the costs of holding unnecessary inventories while finishing a job late results in contractual penalty and loss of customer goodwill

3 1 Previous studies on related models

The study of earliness and tardiness penalties in scheduling models is a relatively recent area of inquiry For many years, scheduling research focused on single performance measures, referred to as *regular* measures that are nondecreasing in job completion times

Most of the literature deals with such *regular* measures as mean flowtime, mean lateness, percentage of jobs tardy, and mean tardiness

The mean tardiness criterion, in particular, has been a standard way of measuring conformance to due dates, although it ignores the consequences of jobs completing early

However, this emphasis has changed with the current interest in Just-In Time (JIT) production, which espouses the notion that earliness, as well as tardiness, should be discouraged [BS90]

In a JIT scheduling environment, jobs that complete early must be held in finished goods inventory until their due date, while jobs that complete after their due dates may cause a customer to shut down operations Therefore, an ideal schedule is one in which all jobs finish exactly on their assigned due dates This can be translated to a scheduling objective in several ways

JIT encompasses a much broader set of principles than just those relating to due dates, but scheduling models with both earhness and tardiness penalties do much to capture the scheduling dimension of a JIT approach

The concept of penalising both earliness and tardiness has spawned a new and rapidly developing line of research in scheduling theory Because the use of both earliness and tardiness penalties gives rise to a nonregular performance measure, it has led to new methodological issues in the design of solution procedures

3 1 1 The E/T model

Virtually all the literature on E/T (earliness and tardiness) problems deals with static scheduling In other words, the set of jobs to be scheduled is known in advance and is available to all schedulers in a multiple machine environment The vast majority of the articles [GK87], [BS90], [C88], [C87], [HP89], [HP91] on E/T problems only deals with single machine models although some single machine results have been extended to parallel machines Let E_j and T_j represent the earliness and tardiness, respectively of job j

Associated with each job is a unit earliness penalty $a_j > 0$ and a unit tardiness penalty $\beta_j > 0$ Job j is also described by a processing time p_j and a due date d_j The basic E/T objective function for a schedule S can be written as f(S) where

$$f(S) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$$

In some formulations of E/T problems the due dates are given while in others they are derived from the optimality function In the simplest models, all jobs have a common due date Prescribing a common due date might represent a situation where several items constitute a single customer's order, or it might reflect an assembly environment in which the components should all be ready at the same time to avoid staging delays

A more general model allows distinct due dates, but in these cases due dates appear to be intrinsically different from solutions to problems with a common due date

Treating due dates as decision variables reflects the practice in some shops of setting due dates internally, as targets to guide the progress of shop floor activities

3.1 2 Minimizing total deviation from a common due date

An important special case in the family of E/T problems involves minimising the sum of absolute deviations of the job completion times from a common due date [K81a], [SH84], [H86], [BCS87] In particular, the objective function can be written as

$$f(S) = \sum_{j=1}^{n} |C_j - d| = \sum_{j=1}^{n} (E_j + T_j)$$

with the understanding that d_j=d

When we write the objective function in this form, it is clear that earliness and tardiness are both penalized at the same rate for all jobs In these cases it is desirable to construct the schedule so that the due date is, in some sense, in the middle of the jobs If d is too small, then it will not be possible to fit enough jobs in front of d, because of the restriction that no job can start before time zero Thus for a given job set we might discover that d is too small, this gives rise to the restricted version of the problem

It can be shown that there exists an optimal solution to the unrestricted problem with the following properties [BS90]

I There is no inserted idle time in the schedule
(If job j immediately follows job i in the schedule the $C_j=C_i+p_j$)

II The optimal schedule is V-shaped (Jobs for which $C_j \leq d$ are sequenced in nonincreasing order of processing time, jobs for which $C_j > d$ are sequenced in nondecreasing order of processing time)

III One job completes precisely at the due date ($C_j=d$ for some j)

IV In an optimal schedule, the bth job in sequence completes at time d, where b is the smallest integer greater than or equal to n/2 In other words, b = n/2 if n is even, and b = (n+1)/2 if n is odd

3 1 3 Parallel machine models

The basic analysis of the unrestricted version has been extended to models involving m parallel machines The multimachine procedure assigns the m longest jobs to different machines Thereafter, the jobs are treated in nonincreasing order of processing times and assigned 2m at a time among the machines After all the jobs are assigned an algorithm is used to sequence the job on each machine

In addition the four key properties apply to the optimal solution of the multimachme model in the form [SA84], [H86]

I On each machine, there is no inserted idle time

II On each machine, the optimal schedule is V-shaped

III On each machine, one job completes at time d

IV The number of jobs assigned to each of the m machines is either [n/m] or [n/m]+1 (where [x] denotes the integer portion of x) Let this number be denoted q Then, on each machine, the bth job in sequence completes at time d, where b is the smallest integer greater than or equal to q/2

3 1 4 Different earliness and tardiness penalties

A generalization of the basic model derives from the notion that earliness and tardiness should be penalized at different rates As noted earlier, α may represent a holding cost while β represents a tardiness penalty These are likely to be different, especially because α tends to be endogenous, while β tends to be exogenous In particular, let

$$f(S) = \sum_{j=1}^{n} (\alpha E_j + \beta T_j)$$

Again there are restricted as well as unrestricted versions of the problem In the unrestricted version an optimal solution has these properties [BCS87]

I There is no inserted idle time

II The optimal schedule is V-shaped

III One job completes at time d

IV In an optimal schedule the b^{th} job in sequence completes at time d, where is the smallest integer greater than or equal to $n\beta/(\alpha+\beta)$

3 1 5 Additional penalties

One way to extend the E/T criterion is to include other performance criteria in which penalties might be incorporated Two such criteria, namely due-date penalty and flowtime penalty, are introduced by Panwalkar, Smith and Seidmann [PSS82] Their model takes the common due date as a decision variable, but their formulation also provides a disincentive for setting a late due date

This structure makes practical sense For example, a firm might offer a due date to its customer during sales negotiations, but have to offer a price reduction if the due date is set too late

Suppose that there is a given parameter d_0 that represents a maximally acceptable due date Consider the following objective function

$$f(S) = \sum_{j=1}^{n} \left[\alpha\, E_j + \beta\, T_j + \gamma\, (d - d_0)^{+}\right]$$

Here, a penalty γ is assessed (for each job) on the difference between the due date selected and d_0, when d is later This penalty provides a disincentive for setting due dates later than the maximally acceptable value Panwalkar, Smith and Seidmann [PSS82] indicate that this problem cannot be solved except by enumerative techniques An exception in the special case $d_0=0$

In addition properties I, II, and III (p39), hold for this problem

Property IV generalizes as follows

IV In an optimal schedule the bth job in sequence completes at time d, where b is the smallest integer greater than or equal to $n(\beta-\gamma)/(\alpha+\beta)$

For a different extension of the E/T model, with d as a decision variable, consider the following objective function

$$f(S) = \sum_{j=1}^{n}\left[\alpha E_j + \beta T_j + \theta C_j\right]$$

Here a penalty is assessed on the completion time (equivalently, the flow time) of job j, thus providing an incentive to turn around orders rapidly

The model contains an additional trade off because the flowtime penalty tends to induce shortest first sequencing whereas the earliness cost induces the reverse sequencing, at the start of the schedule

3 1 6 Nonlinear penalties

In some cases, large deviations from the due date are highly undesirable, and it might be more appropriate to use squared deviations form the common due date as the performance measure Thus, consider the objective function

$$f(S) = \sum_{j=1}^{n}(d - C_j)^2 = \sum_{j=1}^{n}(E_j^2 + T_j^2)$$

This is the quadratic analogue of total absolute deviation Bagchi, Sullivan and Chang [BSC86] show that the unrestricted version of this problem is equivalent to the completion variance problem studied by Eilon and Chowdhury [EC77], Kanet [K81b] and Vani and Raghavachari [VR87]

Eilon and Chowdhury [EC77] propose the first heuristic algorithm for solving the quadratic problem, using adjacent pairwise interchanges of jobs to improve the solution

Kanet [K81b] shows that the problem is equivalent to minimizing the sum of squared differences in job completion times He adapts an algorithm for the absolute deviation problem as a heuristic for the quadratic objective and improves on the Eilon-Chowdhury [EC77] results

Vani and Raghavachari [VR87] investigate the use of all pairwise interchanges, and they obtain improved solutions over the other heuristics at the cost of increased computational time

Bagchi, Chang and Sullivan [BCS87] also examine the general case in which earliness and tardiness penalties differ

$$f(s) = \sum_{j=1}^{n} (\alpha\, E_j^2 + \beta\, T_j^2)$$

They develop dominance properties and incorporate them into a search procedure to solve the problem, however their approach remains essentially an enumerative one

3 1 7 Job dependent earliness and tardiness penalties

An obvious direction for generalization is to permit each job to have its own penalties α_j and β_j Specifically the objective function takes the form

$$f(S) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$$

When $\alpha_j = \beta_j$, the tardiness penalty matches the earliness penalty for any particular job, but the penalties may differ among jobs The unrestricted version of this problem has been examined by Bagchi [B85], Cheng [C87], Quaddus [Q87], Bector, Gupta and Gupta [BGG88],and Hall and Posner [HP89]

Bagchi [B85] considers the case in which $a_j = a\ p_j$ He proves some dominance properties that might accelerate a solution procedure Bector, Gupta and Gupta [BGG88] present a linear programming perspective on these same results Hall and Posner [HP89] prove some dominance properties that provide necessary conditions for an optimal sequence Their most significant result is a proof that the unrestricted version of he problem is NP-complete

They proceed to develop a dynamic programming algorithm, which they show to be pseudopolynomial

Furthermore, they demonstrate the computational effectiveness of their algorithm by attacking problems that contain hundreds of jobs and by obtaining optimal solutions with modest run times

Quaddus [Q88] considers the general case in which $\alpha_j \neq \beta_j$ and also includes a due date penalty γ_j but deals only with the selection of a due date

However it is easy to show that Property I (p39) holds, and Property II (p39) takes the form

II The optimal schedule is V-shaped jobs in B are sequenced in non-increasing order of the ratio p_j / α_j and jobs in A are sequenced in non-decreasing order of the ratio p_j / β_j

In addition Property III (p39) holds and the general form of Property IV specifies a necessary condition for b as

IV In an optimal schedule the bth job in sequence completes at time d, where b is the smallest integer satisfying the inequality

$$\sum_{j=1}^{b}(\alpha_j + \beta_j) \geq \sum_{j=1}^{n}(\beta_j - \gamma_j)$$

where the subscript j denotes the jth job in sequence

3 1 8 Due date tolerances

A more general representation allows the penalty to be zero if the completion time is close enough to the due date, where close enough is specified by a given tolerance For job j to avoid penalties, its completion time must fall in the interval from $d-u_j$ to $d+v_j$ This interval could be interpreted as the length of a time bucket in an MRP system Cheng [C88] analyzes a special case in which the criterion is total absolute deviation and all u_j and v_j are identical He imposes an unusual assumption

Although the model prescribes no penalty on a job that completes within its tolerance interval around the due date, other jobs have earliness and tardiness calculated from the due date rather than from the end of the tolerance interval This gives rise to the discontinuous penalty function shown below

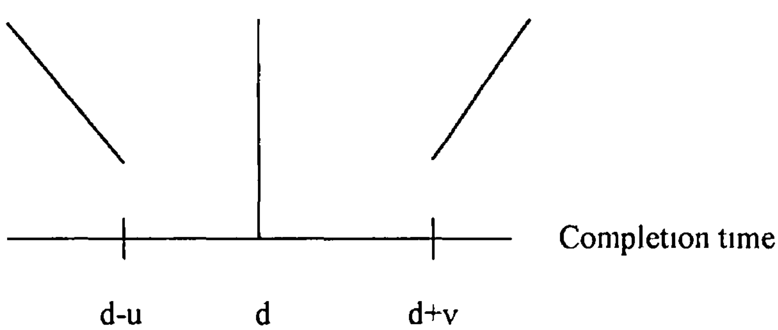

Figure 3 Discontinuous penalty function

Consider the more conventional, and consistent assumption that, for job j, earliness or tardiness is measured only from the end of the tolerance interval

$$E_j = (d-C_j-u_j)^+$$

$$T_j = (C_j-d-v_j)$$

and $$f(S) = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$$

This gives rise to the continuous penalty function shown below

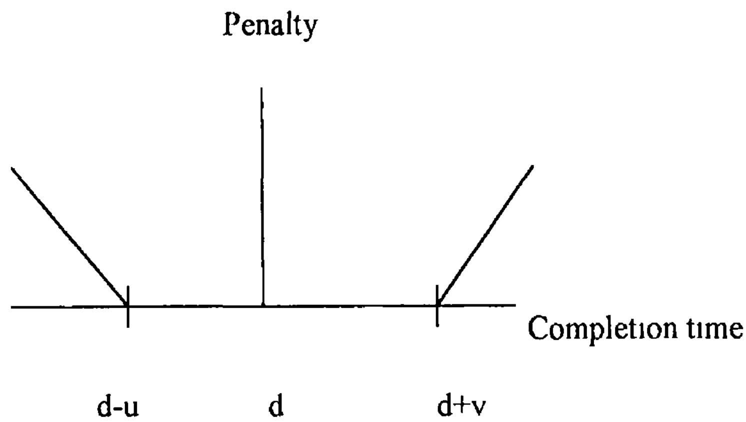

Figure 4 Continuous penalty function

The tolerance is also assumed to be relatively small compared to the processing times in the job set Formally, the requirement is that at most one job can avoid penalty costs or

$p_j - v_j - u_j > 0$ for all pairs of jobs (i,j)

Notice that the models previously discussed can be viewed as the special case in which $u_j = v_j = 0$

In the tolerance model, Properties I and II (p39) continue to hold The generalization of Property III states that there will be one job that incurs no penalty in the optimal solution, we shall treat this as job b The generalization of Property IV provides a necessary condition for b in an optimal sequence

Property III (generalized)

In an optimal schedule some job j completes either at $d-u_j$ or at $d+v_j$

Property IV (generalized)

In an optimal schedule let b denote the number of jobs that incur no tardiness penalty Then the completion time of job j satisfies the conditions

$$C_b = d-u_b \quad \text{if} \quad \sum_{i<b}\alpha_i < \sum_{i\geq b}\beta_i \quad \textit{and} \quad \sum_{i<b}\alpha_i \geq \sum_{i>b}\beta_i$$

$$C_b = d+u_b \quad \text{if} \quad \sum_{i<b}\alpha_i < \sum_{i>b}\beta_i \quad \textit{and} \quad \sum_{i\leq b}\alpha_i \geq \sum_{i>b}\beta_i$$

Thus, for the tolerance model

- Properties I and II (p39) apply
- In the unrestricted version of the problem Properties III(generalized) and IV (generalized) apply
- Determining whether a given problem is restricted requires solving the unrestricted version, with the due date as a decision
- Constructing the optimal solution requires a matching of coefficients and processing times when $\alpha_j = \alpha$ and $\beta_j = \beta$ and the problem is unrestricted

Otherwise the solution requires an enumeration of V-shaped sequences, aided to Property IV(generalized) which identifies those V-shaped sequences that are candidates for optimality

3.1.9 The minimax criterion

One other tolerance model that consists of minimizing the maximum penalty, where penalties are assessed on earliness or tardiness has been studied. In other words the objective function is [S77]

$$f(S) = \min_j\{\max[\alpha(E_j),\beta(T_j)]\}$$

where $\alpha(X)$ and $\beta(X)$ are convex functions of earliness and tardiness, and distinct due dates are permitted.

3.1.10 Distinct due dates

The general E/T model has different due dates in the job set. This feature tends to make it more difficult to determine a minimum cost schedule than in the problems discussed so far. However, if the due dates are treated as decision variables, the problem turns out to be relatively simple. The objective function has the form

$$f(S) = \sum_{j=1}^{n}\left[\alpha\, E_j + \beta\, T_j + \gamma\,(d - d_0)^+\right]$$

In this model, Properties I and II (p39) do not hold, the optimal sequence may not be V-shaped, and inserted idle time may be desirable. The search for an optimal schedule can, however, be decomposed into two subproblems: finding a good job sequence and scheduling inserted idle time.

3 1 11 Job deadlines

A related model introduces *deadlines* rather than due dates [B87] Whereas due dates may be violated at the cost of tardiness, deadlines must be met and cannot be violated Thus for example, if the makespan exceeds the maximum allowable deadline, then the problem is considered infeasible However we can also view such models as E/T models with infinite β_j, and thus special cases of the problem considered above A more general objective is to minimize total weighted earliness

The objective function can be stated formally as

$$\text{Min } f(D,\sigma) = \sum_j n_j \theta_j C_j + \sum_j \sum_i (\alpha_j E_{ij} + \beta_j T_{ij})$$

where

n_j = number of jobs in customer order j

θ_j = lead-time penalty per unit time for each job in customer order j

C_j = completion time of the last job in customer order j

T_{ij} = tardiness of job i in customer order j

E_{ij} = earliness of job i in customer order j

α_j = unit earliness penalty for customer order j

β_j = unit tardiness penalty for customer order j

and D is the vector of the due dates for the customer orders

CHAPTER 4

A MIXED INTEGER PROGRAMMING FORMULATION OF SCHEDULING PROBLEMS

4.0 Sciconic

In this section we are going to present the MIP (Mixed Integer Programming) formulation of our problem along with the results that we obtained for a specific instance of our model, using SCICONIC an algorithmically advanced Mathematical Programming package developed by SCICON Its purpose is to provide both technical and non-technical users with a convenient and cost-effective way to solve linear, integer and non-linear programming problems Mathematical programming (MP) is a rapidly advancing field, and SCICONIC has been designed around advanced algorithms and techniques Further developments are continually being made, especially in robustness and the speed of solution for large linear and mixed integer problems

Mathematical programming (MP) has a wide variety of applications in the petroleum, chemical and manufacturing industries, transport agriculture and many more It can be used for a variety of purposes, from providing an optimum solution to an established problem to providing a frame work for collecting and evaluating all of the relevant data and their consequences Completely new models can be built in order to gain greater understanding of a hypothetical situation, while established models can be run routinely many times a day to guide the operation of a manufacturing process

In general terms, Mathematical Programming is concerned with the best way to allocate scarce resources to alternative activities [W78] lists applications under the following headings

- The Petroleum Industry
- The Chemical Industry
- Manufacturing
- Transport
- Finance
- Agriculture
- Health
- Mining
- Manpower Planning
- Food
- Energy
- Pulp and Paper
- Advertising
- Defence
- Other applications

It is important to realise why mathematical programming applications have been successful Firstly they give true optimum solutions to a well-defined problem Secondly, the concepts of Mathematical programming – the quantification of the objectives and the set of all possible ways of achieving these objectives – provide a framework for thinking about all the relevant data, an occasion for collecting them and the ability to compute the consequences of these data Often the only way to achieve a realistic set of data is to show the people who collected them the consequences of their initial estimates of the data values

It is useful to distinguish between established and new mathematical programming models An established model is run from time to time with updated input data as part of some operational decision – making routine The purpose is then to suggest a specific course of action to management, and the suggestion will usually be accepted A new model may also be used in this way, but is more often used to gain greater understanding of the situation The model may be run under a variety of assumptions that lead to different conclusions, and the model itself will not suggest which set of assumptions is most appropriate

During the model development and data gathering phase, one must therefore be prepared to make many optimisation calculations which the analyst will show to management and say *"This is what the model now recommends Does it look sensible, and if not why not ?"* Neither the analyst nor the manager should accept the recommendations unless they can be explained qualitatively as the natural consequences of physical and economic assumptions This can be paraphrased by saying that one should only trust the model if the results are obvious This may suggest that the model is of no real use but this is not so, because many things are obvious once someone has pointed them out, when they were not at all obvious beforehand

One difficulty with large-scale mathematical programming models is that the details of the formulation can become obscure, and changes are then hazardous So we need a systematic approach to documentation It is natural to base this on compactness of an algebraic formulation

Two important points that have to be mentioned are the following

1 Practical linear programming formulations can all too easily require hundreds of constraints and thousands of variables, while

2 The algebraic formulation is precise and often compact

Mathematically, MP is about finding the maximum or minimum value of a function of several variables given that the variables have to satisfy a number of constraints, which are limits on the values of functions of the variables

The stages involved running mathematical programming models are

- Express the problem as an MP matrix
- Find the optimum solution
- Interpret the solution

An MP code such as SCICONIC will do the second stage It takes the matrix in standard (MPS) format (the MPS code and the output that we get using SCICONIC can be found in Appendix C), finds the optimal solution and writes it out to a solution file

SCICONIC expects a problem to be presented in the form of an industry- standard MPS format matrix file Creating an MPS matrix by hand in an editor is a slow and error prone task, even for very small problems

4 1 Mixed Integer Programming (MIP) formulation

A large number of MIP formulations have been proposed by a number of authors [F82] A new MIP formulation that has been recently used by [AC91] is presented here Keeping the notation defined above, the problem can be formulated as follows

Minimize C_{max}

Subject to $\forall\, i \in X,\ t_i \geq 0,$

$\forall\, i \in X,\ C_{max} \geq t_i + p_i$

$\forall\, (i,j) \in U,\ t_j \geq t_i + p_i$

$\forall\, [i,j] \in D,\ t_j \geq t_i + p_i \text{ or } t_i \geq t_j + p_j$

This disjunctive programming problem leads to the following MIP formulation by introducing a binary variable y_{ij} and setting the new constraints

$$\forall\ [i,j] \in D,\ t_i \geq t_j + p_j - K y_{ij},\ t_j \geq t_i + p_i - K(1 - y_{ij})$$

$$\forall\ [i,j] \in D,\ y_{ij} \in \{0,1\},$$

where K is some large constant, and $y_{ij}=1$ if and only if i is scheduled before j, and 0 otherwise

4 2 An example of the (MIP) formulation

Three jobs A,B and C are to be processed on a single machine

Job A is processed on the machine for p_A hours, job B is processed on the machine for p_B hours and finally job C is processed on the machine for p_C hours

The machine can work only one job at a time and no preemption is allowed We also assume that we have two job families (f=2) which are defined as follows family1 $f_1=\{A,C\}$ family2 $f_2=\{B\}$

While s_i , i=1,2 denotes the setup time required in order to process a job in family i, following a job in some other family No setup is required between jobs from the same family

All jobs require the same due-date that is denoted d that means that is desirable to finish jobs in no more than d hours

Formulation

Let X_A denote the time (measured from zero datum) when the processing of job A is started on the machine Similarly X_B and X_C are defined

The first set of pertinent constraints is the non-interference constraints, which guarantee that machine work on no more than one job at a time

For instance the machine can work on either job A or B, or C at any given time

This is equivalent to the statement that either job A precedes job B on the machine or vice versa

Thus we have an "either-or" type constraint for non-interference on the machine given by

$$X_A+p_A+s_2\leq X_B$$

or

$$X_B+p_B+s_1\leq X_A$$

With the help of a binary integer variable, the "either-or" constraint can be reduced to the following two constraints

$$X_A+p_A+s_2-X_B\leq M\delta_1 \qquad (1)$$

$$X_B+p_B+s_1-X_A\leq M(1-\delta_1) \qquad (2)$$

where $0\leq \delta_1\leq 1$, δ_1 is integer, and M is a large positive number Note that when $\delta_1=1$, the first constraint becomes $X_A+p_A+s_2-X_B\leq M$ and is inactive, while the second constraint reduces to $X_B+p_B+s_1-X_A\leq 0$ implying job B precedes job A on the machine On the other hand, when $\delta_1=0$, the first constraint becomes

$X_A+p_A+s_2-X_B\leq 0$ implying that job A precedes job B, while the second constraint becomes $X_B+p_B+s_1-X_A\leq M$ and is inactive

Thus with the help of the binary integer variable both possibilities are simultaneously included in the problem

Because the single machine can process any of the three jobs A,B and C at any time we obtain

$$X_A+p_A\leq X_C$$

or

$$X_C+p_C\leq X_A$$

(the factor s_i is missing because jobs A and C belong to the same family f_1)

With the help of the binary integer variable δ_2 we obtain

$$X_A+p_A-X_C\leq M\delta_2 \qquad (3)$$

$$X_C+p_C-X_A\leq M(1-\delta_2) \qquad (4)$$

$$X_B+p_B+s_1\leq X_C$$

or

$$X_C+p_C+s_2\leq X_B$$

With the help of the binary integer variable δ_3 we obtain

$$X_B+p_B+s_1-X_C\leq M\delta_3 \qquad (5)$$

$$X_C+p_C+s_2-X_B\leq M(1-\delta_3) \qquad (6)$$

Where $0\leq \delta_1 \leq 1$, $0\leq \delta_2 \leq 1$, $0\leq \delta_3 \leq 1$, δ_1, δ_2 and δ_3 are integers

Because of using due-date tolerances which means that we allow the penalty to be zero if the completion time of job j falls in the interval $(d-u_j\ ,\ d+v_j)$ the due-date constraints for jobs A, B and C become

$$d-u_A\leq X_A+p_A\leq d+v_A \qquad (7)$$

$$d-u_B\leq X_B+p_B \leq d+v_B \qquad (8)$$

$$d-u_C\leq X_C+p_C \leq d+v_C \qquad (9)$$

Constraints (7), (8), (9) mean that jobs A, B and C are allowed to be completed in the intervals

$(d-u_i\ ,\ d+v_i)$, where i=A, B, C respectively

We know in general that for job j earliness is defined as $E_j=\max\{0,d-C_j-u_j\}$ (where C_j is the completion time of job j)

Equivalently in our problem for job A we obtain $E_A=\max\{0,d-(X_A+p_A)-u_A\}$

This function is equivalent to the following constraints

$E_A \geq 0$ (10)

$E_A \geq d - X_A - p_A - u_A$ (11)

Similarly for jobs B and C we obtain

$E_B \geq 0$ (12)

$E_B \geq d - X_B - p_B - u_B$ (13)

$E_C \geq 0$ (14)

$E_C \geq d - X_C - p_C - u_C$ (15)

In general for job j tardiness is defined as $T_j = \max\{0, C_j - d - v_j\}$

Equivalently for job A we obtain $T_A = \max\{0, (X_A + p_A) - d - v_A\}$

This function is equivalent to the following constraints

$T_A \geq 0$ (16)

$T_A \geq X_A + p_A - d - v_A$ (17)

Similarly for jobs B and C we obtain,

$T_B \geq 0$ (18)

$T_B \geq X_B + p_B - d - v_B$ (19)

$$T_C \geq 0 \qquad (20)$$

$$T_C \geq X_C + p_C - d - u_C \qquad (21)$$

The objective is to find S^* satisfying

$$F(S^*) = \min_{S \in \Pi} \{F(S)\}$$

where

$$F(S) = \sum_{j=1}^{n} [a_j E_j(S) + \beta_j T_j(S)]$$

and Π denotes the set of all feasible schedules

4 3 Conclusions

In this chapter we present the MIP (Mixed Integer Programming) formulation of our model along with an example of the MIP formulation, considering three jobs that belong to two families

In Appendix C we present the results that we obtained using SCICONIC, for a specific case instance considering a set of four independent jobs with processing times $t_1 = 1$, $t_2 = 3$, $t_3 = 6$, and $t_4 = 10$ for jobs 1,2,3,4 respectively (the same example as in paragraph 5 1 3)

We can notice in Appendix C that the solution that we get from SCICONIC is equal to the optimal solution that we can get using full enumeration

SCICONIC might have been used to provide tight bounds on the quality of heuristic solutions that will be presented

Unfortunately because SCICONIC expects a problem to be presented in the form of an industry standard MPS format matrix file Creating an MPS matrix by hand in an editor is a slow and error prone task, even for very small problems and therefore we could not use SCICONIC in order to provide tight bounds on the quality of the solutions that will be presented

CHAPTER 5

AN ALGORITHM FOR SCHEDULING GROUPS OF JOBS ON A SINGLE MACHINE

5.0 Introduction

In this chapter we develop an algorithm (JGA) for scheduling groups of jobs on single machine in order to minimise an objective function

We also illustrate the operation of the algorithm using a specific example Three Lemmas are presented to illustrate the use of the results to determine the optimal solution to the due-date determination and sequencing problem

5 1 Scheduling independent jobs on a single machine

Although we are concerned with the optimal sequencing of a set of group of jobs, to minimise a penalty of deviation from the desired due-dates, (a problem which is coupled with the optimal assignment of due-dates to the set of jobs to be processed by a single machine), in this section we consider the case where the jobs are independent (they do not belong to any family) [C88]

Let N be the set of n independent jobs to be processed on a single machine Each job requires t_i time units of processing on the machine, $\forall\ i \in N$

The common due-date assignment method is employed to assign due-dates to jobs

Thus, each job i is assigned a due-date $d_i = r_i + k$ where r_i is the ready time of job i and k is a common flow allowance, $\forall i \in N$

While it is true that that there will be penalties for failing to complete a job on its due-date, in practice such a penalty will not occur if the deviation of job completion time form the due-date is sufficiently small

Thus the jobs are given a completion time deviation allowance α such that there will be no penalties if the completion time of job i is within the time interval $(d_i - \alpha, d_i + \alpha)$, $\forall i \in N$

The basic assumptions about the problem model are as follows

- The job processing times t_i $\forall i \in N$ are known and deterministic
- The jobs are available for processing at the same time, i e $r_i = 0$ $\forall i \in N$
- There is a single machine available, which can only process the jobs one at a time
- Job splitting and preemption are not allowed
- The completion time deviation allowance α is sufficiently small and satisfies the condition $2\alpha < \min \{t_i\}$ $\forall i \in N$

We define

$$E_{[i]} = \max\{0, C_{[i]} - k\}$$

$$T_{[i]} = \max\{0, k - C_{[i]}\}$$

$$C_{[i]} = \sum_{j=1}^{i} t_j$$

Let Π be the set of all possible job sequences and σ be an arbitrary sequence

Let the subscript [i] denote the job in position i of σ

Let t_i denote the processing time of job i and $t_{[i]}$ denote the processing time of job in i-th position of a sequence σ

Let $E_{[i]}$, $T_{[i]}$ and $C_{[i]}$ be the earliness, tardiness and completion time of the ith job in σ respectively, then the objective is to minimise a penalty function of missing due-dates expressed as

$$f(k,\sigma) = \sum_{i=1}^{n} \{E_{[i]}\, U(E_{[i]} - \alpha) + T_{[i]}\, U(T_{[i]} - \alpha)\} \tag{1}$$

Here U(x-c) is the unit step function defined as

$$U(x-c) = \begin{Bmatrix} 0 \text{ if } x \le c \\ 1 \text{ if } x > c \end{Bmatrix}$$

5 1 1 The optimal due-date

In this section we present two lemmas [C88] which are used to help determine the optimal value of the common flow allowance k^*

Lemma 1. For a given job sequence σ the optimal due-date must equal one of the job completion time minus the completion time deviation allowance α i e $k^* = C_{[i]} - \alpha$, $\exists\ [i] \in N$

Proof of Lemma 1

Let k be an arbitrary chosen common due-date (i e $C_{[i-1]} < k < C_{[i]}$, i=1,2, ,n-1) which does not have the property as stated in Lemma 1 Then in the form of a Gantt-chart, k will be like the following

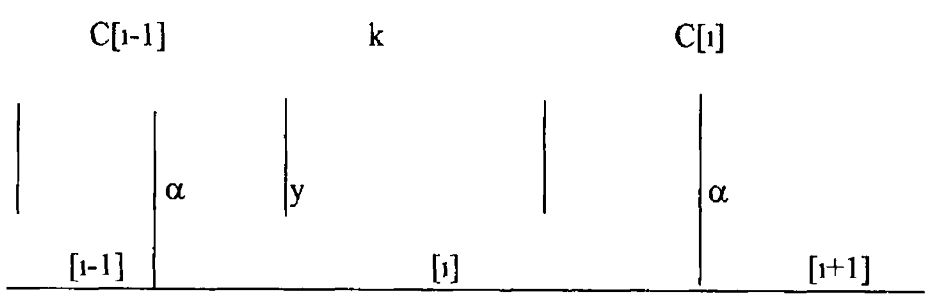

Figure5 Gantt Chart for Lemma 1

Now shifting k to the right side so that it equals $C_{[i]}$-α causes the following change in penalty

$$\Delta P_R \;=\; (i-1)(t_{[i]}-y-\alpha)-(n-i+1)(t_{[i]}-y-\alpha) \;=\; (2i-n-2)(t_{[i]}-y-\alpha) \qquad (2)$$

Similarly, shifting k to the left side so that it equals $C_{[i-1]}$- α gives rise to the following change in penalty

$$\Delta P_L \;=\; (n-i+1)(y+\alpha) \;-\; (i-1)(y+\alpha)= \;(n-2i+2)(y+\alpha) \qquad (3)$$

It is evident from (2) and (3) that

$$\Delta P_R \quad \leq 0 \;\; if \;\; i \leq \frac{n}{2}+1$$

$$\Delta P_L \quad \leq 0 \;\; if \;\; i \geq \frac{n}{2}+1$$

Thus for any given k, we can perform an appropriate left or right shift depending on the value of k so that a reduced or equal penalty value can be achieved It follows that the optimal due-date must satisfy the condition that $k^{*}= C_{[i]}$ -α, $\exists$ $[i] \in N$

<u>**Lemma 2**</u> For a given job sequence σ, the optimal due-date is $k^* = C_{[r]} - \alpha$

where r is such that

$$r = \begin{cases} (n+1)/2 & if\ n\ is\ odd \\ n/2+1 & if\ n\ is\ even \end{cases}$$

Proof of Lemma 2

From Lemma 1 we know that the optimal due-date is $k^* = C_{[r]} - \alpha$, $\exists$ $[r] \in N$ Let $k^+ = C_{[r+1]} - \alpha$ and $k^- = C_{[r-1]} - \alpha$ Since k^* is optimal the following conditions must be satisfied

$$f(k^*,\sigma) - f(k^+,\sigma) \leq 0 \qquad (4)$$

and

$$f(k^*,\sigma) - f(k^-,\sigma) \leq 0 \qquad (5)$$

Clearly

$$f(k^*,\sigma) = \sum_{i=1}^{r-1}(C_{[r]} - \alpha - C_{[i]}) + \sum_{i=r+1}^{n}(C_{[i]} - C_{[r]} + \alpha) \qquad (6)$$

$$f(k^+,\sigma) = \sum_{i=1}^{r}(C_{[r+1]} - \alpha - C_{[i]}) + \sum_{i=r+2}^{n}(C_{[i]} - C_{[r+1]} + \alpha) \quad (7)$$

Substituting (6) and (7) into (4), we obtain

$$(r-1)\, t_{[r+1]} + (t_{[r+1]} - \alpha) - (t_{[r+1]} + \alpha) - [(n-r-1)t_{[r+1]}] \geq 0$$

or

$$r \geq n / 2 + \alpha / t_{[r+1]} \qquad (8)$$

Also

$$f(k^-,\sigma) = \sum_{i=1}^{r-2}(C_{[r-1]} - \alpha - C_{[i]}) + \sum_{i=r}^{n}(C_{[i]} - C_{[r-1]} + \alpha) \qquad (9)$$

Similarly, substituting (6) and (9) into (5), we obtain

$$-(r-2)t_{[r]}-(t_{[r]}+\alpha)+(t_{[r]}-\alpha)+(n-r)t_{[r]} \geq 0$$

or

$$r \leq n/2 + 1 + \alpha/t_{[r]} \qquad (10)$$

Since it has been assumed that $2\alpha \leq \min \{t_i\}\ \forall\ i \in N$ and r must be non-negative integer, it is clear from (8) and (10) that

$$r = \begin{cases} (n+1)/2 & \text{if } n \text{ is odd} \\ n/2+1 & \text{if } n \text{ is even} \end{cases}$$

and the proof is complete

It is interesting to note from Lemma 2 that for a set of jobs, the optimal due-date is an explicit function of the size of the job-set n and can be uniquely determined for a given job sequence

5 1 2 The optimal sequence

Once the optimal due-date value k* is determined, we can use the following lemma [C88] to find the optimal job sequence σ*

Lemma 3 For a given optimal due-date $k^* = C_{[r]} - \alpha$ as determined from Lemma 2, there exists an optimal job sequence that has the property

$$t_{[j]} \geq t_{[n+1-j]} \geq t_{[j+1]},\ j = 1,2, \ldots ,r-1$$

Proof of Lemma 3

Let σ_1 be an optimal job sequence that does not have the property described above

There must exist a pair of jobs p and q in position m and n-m+1, m$\leq$ r-1, respectively that $t_p < t_q$ Now we construct a new sequence σ_2 in which p and q are interchanged in position while all other jobs are in the same position as in σ_1 It follows from Lemma 2 that

$$k_1^* = C_{[m-1]} + t_p + \sum_{i=m+1}^{r} t_{[i]} - \alpha \qquad (11)$$

$$k_2^* = C_{[m-1]} + t_q + \sum_{i=m+1}^{r} t_{[i]} - \alpha \qquad (12)$$

Observe that $k_1^* - k_2^* = t_q - t_p > 0$ (13)

$$\begin{aligned} f(k_1^*,\sigma_1) &= \sum_{i=1}^{r-1}(k_1^* - C_{[i]}) + \sum_{i=r+1}^{n}(C_{[i]} - k_1^*) = \sum_{i=1}^{m-1}(k_1^* - C_{[i]}) + k_1^* - (C_{[m-1]} + t_p) \\ &+ \sum_{i=m+1}^{r-1}\left\{k_1^* - (C_{[m-1]} + t_p + \sum_{j=m+1}^{i} t_{[j]})\right\} + \sum_{i=r+1}^{n-m}\left\{(C_{[m-1]} + t_p + \sum_{j=m+1}^{i} t_{[j]}) - k_1^*\right\} \\ &+ \sum_{i=n-m+1}^{n}(C_{[i]} - k_1^*) \qquad (14) \end{aligned}$$

$$\begin{aligned} f(k_2^*,\sigma_2) &= \sum_{i=1}^{r-1}(k_2^* - C_{[i]}) + \sum_{i=r+1}^{n}(C_{[i]} - k_2^*) = \sum_{i=1}^{m-1}(k_2^* - C_{[i]}) - k_2^* - (C_{[m-1]} + t_q) \\ &+ \sum_{i=m+1}^{r-1}\left\{k_2^* - (C_{[m-1]} + t_q + \sum_{j=m+1}^{i} t_{[j]})\right\} + \sum_{i=r+1}^{n-m}\left\{(C_{[m-1]} + t_q + \sum_{j=m+1}^{i} t_{[j]}) - k_2^*\right\} \\ &+ \sum_{i=n-m+1}^{n}(C_{[i]} - k_2^*) \qquad (15) \end{aligned}$$

Subtracting (15) from (14) and simplifying using (13) we obtain

$$f(k_1^*,\sigma_1)-f(k_2^*,\sigma_{21})=t_q-t_p>0$$

Thus the interchange of p and q reduces the penalty value Therefore any sequence that does not have the property $t_{[j]} \geq t_{[n+1\ j]} \geq t_{[j+1]}$, $j = 1,2, \ ,r-1$ can be improved by such an interchange of pairs of p and q It follows that the sequence having the property itself must be optimal

5 1 3 Numerical example for a set of independent jobs

A set of four indpendent jobs is given with $t_1=1$, $t_2= 3$, $t_3= 6$, and $t_4= 10$ The completion time deviation allowance is $\alpha=0\ 45$

According to Lemma 2 we know that

$$r=\begin{cases}(n+1)/2 & \text{if } n \text{ is odd}\\ n/2+1 & \text{if } n \text{ is even}\end{cases}$$

thus r=3, so $k^* = C_{[3]} - \alpha$ and the optimal sequence σ^* can be constructed using the Lemma 3 as follows

We know that $t_{[j]} \geq t_{[n+1\ j]} \geq t_{[j+1]}$, $j = 1,2, \ ,r-1$

in our case r=3 therefore we have that $t_{[j]} \geq t_{[n+1-j]} \geq t_{[j+1]}$, $j = 1,2$

Thus

$t_{[1]} \geq t_{[4]} \geq t_{[2]}$ (for j=1) (a) and $t_{[2]} \geq t_{[3]} \geq t_{[3]}$ (for j=2) (b)

Combining (a) and (b) we get the following formula $t_{[1]} \geq t_{[4]} \geq t_{[2]} \geq t_{[3]}$ That means that the job with the biggest processing time is going to be processed first Afterwards the job with the second biggest processing time is going to be processed in the fourth place, then the job with the third processing time is going to be processed in the second place and finally the job with the smallest processing time is going to be processed last

Therefore the order in which the jobs are going to be processed is the following 4-2-1-3, $k^* = C_{[3]}-\alpha = 13\ 55$ and the minimum penalty value is 10 55

For this problem there are 4!=24 possible different sequences and the details of each individual sequence is shown in table 1

Table1 Complete enumeration of the set job

σ	R	$k^*=C_{[r]}-\alpha$	$f(k^*,\sigma)$
1-2-3-4	3	9.55	24.55
1-2-4-3	3	13.55	28.55
1-3-2-4	3	9.55	21.55
1-3-4-2	3	16.55	28.55
1-4-2-3	3	13.55	21.55
1-4-3-2	3	9.55	24.55
2-1-3-4	3	9.55	22.55
2-1-4-3	3	13.55	26.55
2-3-1-4	3	9.55	17.55
2-3-4-1	3	18.55	26.55
2-4-1-3	3	9.55	16.55
2-4-3-1	3	18.55	22.55
3-1-2-4	3	9.55	16.55
3-1-4-2	3	16.55	23.55
3-2-1-4	3	9.55	14.55
3-2-4-1	3	18.55	23.55
3-4-1-2	3	16.55	14.55
3-4-2-1	3	18.55	16.55
4-1-2-3	3	13.55	12.55
4-1-3-2	3	16.55	15.55
4-2-1-3	3	13.55	10 55♦
4-2-3-1	3	18.55	15.55
4-3-1-2	3	16.55	10 55♦
4-3-2-1	3	18.55	12.55

♦ Optimal Sequence σ

5.2 Job Grouping Algorithm

In this section we present the Job Grouping Algorithm (JGA) that we developed in order to schedule a number of families (where each family consists of a number of jobs) on a single machine

Our algorithm considers a f-families, n-job, single machine scheduling problem with common due-dates

Suppose that jobs each belong to a particular family, where jobs in a family tend to be similar in some way, such as their required tooling or their container size As a result of this similarity, a job does not need a set-up when following another job from the same family, but a known "family set-up time" is required when a job follows a member of some other family This is called **family scheduling model**

In the family scheduling model, a machine is assumed capable of processing at most one job at a time We use the pair (i,j) to refer to job j of family i

We let f denote the number of families, n the number of jobs, and n_i the number of jobs belonging to family i

In addition $t_{i,j}$ and $w_{i,j}$ denotes the processing time and weight of job (i,j) Thus $n_1 + n_2 + \quad + n_f = n$ In addition, s_i denotes the setup time required to process a job in family i following a job in some other family

If a job follows a member of the same family, then its setup time is zero otherwise its setup time is s_i, the family setup time

The jobs are given a completion time deviation allowance α such that there will be no penalties if the completion time of job i is within the time interval

$(d_i - \alpha, d_i + \alpha)$, $\forall i \in N$

A simplifying assumption for family scheduling is the requirement of precisely f setups in the schedule one for each family (**GT** assumption)

Each family is treated as a single entity, or composite job with processing time

$$p_i = \sum_{j=1}^{n} t_{i,j} \text{ and weight } w_i = \sum_{j=1}^{n} w_{i,j} \text{, and } w_{i,j} = 1 \ \forall \ i, j$$

In addition let $l_i = (s_i + p_i) / w_i = (s_i + p_i) / n_i$ denote the family factor of family i

Let l_i denote the family factor of family i and $l_{[i]}$ denote the family factor of family in i-th position of a schedule σ

This factor is the basis of the proposed algorithm, and actually shows the "importance" of each family Therefore if $l_i > l_j$ and $i \neq j$ then we can say that family i is more "important" in a way than family j (that does not mean that family i is necessarily going to be scheduled earlier than family j)

Applying the proposed algorithm we observe that the family with the largest family factor is always scheduled first in the optimal schedule σ^*

The basic assumptions about the problem model are as follows

- The job processing times $t_{i,j}$ (for all i,j) are known and deterministic
- The jobs are available for processing at the same time, i e $r_{i,j} = 0$ (for all i,j)
- There is a single machine available, which can only process the jobs one at a time
- Job splitting and preemption are not allowed
- The completion time deviation allowance α is sufficiently small and satisfies the condition $2\alpha < \min \{t_{i,j}\}$ for all i,j

The input data for this algorithm are the following

- The number of families that have to be scheduled on the single machine
- The number of jobs in each family
- The processing time for each job in each family
- The completion time deviation allowance α
- The setup time s_i for family i

JGA ALGORITHM

STEP 1 Compute the family factor $l_i = (s_i + p_i) / w_i = (s_i + p_i) / n_i$

(because $w_{i,j} = 1$ for all i,j) for each family i $i = 1,2, \ldots, f$

STEP 2: Compute the value of m where

$$m = \begin{cases} (f+1)/2 & \text{if } f \text{ is odd} \\ f/2+1 & \text{if } f \text{ is even} \end{cases}$$

STEP 3: Compute the value of r where

$$r = \begin{cases} (n+1)/2 & \text{if } n \text{ is odd} \\ n/2+1 & \text{if } n \text{ is even} \end{cases}$$

STEP 4 Find the optimal sequence of families σ^* using the following property

$l_{[j]} \geq l_{[f+1-j]} \geq l_{[j+1]}$, $j = 1,2, \ldots, m-1$

STEP 5 The optimal due date k^* is determined as $k^* = C_{[r]} - \alpha$

STEP 6. The value of the objective function is $f(k^*, \sigma^*)$

5 2 1 Numerical example

In this section we present a numerical example of our algorithm for a specific case instance

In this example we have f = 2 families which we denote by F1 and F2

Each family consists of two jobs (therefore n = 4, n_1 = 2, n_2 = 2) and the processing time for each job is $t_{1\,1}$ = 1, $t_{1\,2}$ = 3, $t_{2,1}$ = 6 and $t_{2,2}$ = 10 The setup times are s_1 = 0 5 and s_2 = 0 1 respectively

Thus F1 = {(1,1), (1,2)} and F2 = {(2,1),(2,2)}

Applying the **first step** of our algorithm we must first compute the family factors

$l_i = (s_i+p_i) / w_i$, i = 1,2

Therefore $l_1 = (s_1+p_1) / n_1$ = (0 5+4) / 2 = 2 25 and

$l_2 = (s_2+p_2) / 2$ = (0 1+16) / 2 = 8 05

Applying the **second step** of our algorithm we compute the value of m where m

$$m = \begin{cases} (f+1)/2 & \textit{if } f \textit{ is odd} \\ f/2+1 & \textit{if } f \textit{ is even} \end{cases}$$

thus m = 2

Applying the **third step** of our algorithm we know that

$$r = \begin{cases} (n+1)/2 & \textit{if } n \textit{ is odd} \\ n/2+1 & \textit{if } n \textit{ is even} \end{cases}$$

thus r=3, so k* = $C_{[3]}$ - α and the optimal sequence σ* can be constructed using the **fourth step** of our algorithm as follows

We know that $l_{[j]} \geq l_{[f+1\ j]} \geq l_{[j+1]}$, j = 1,2, ,m-1

In our case m = 2 therefore we have that $l_{[j]} \geq l_{[f+1\ j]} \geq l_{[j+1]}$, $j = 1$

Thus

$$l_{[1]} \geq l_{[2]} \geq l_{[2]} \text{ (for } j=1)$$

That means that the family with the largest family factor is going to be processed first in the optimal sequence σ^*

Therefore the order in which the groups of jobs are going to be processed is the following

F2-F1

This sequence of the groups of jobs is the same with the following sequence of jobs (2,1) - (2,2) - (1,1) - (1,2)

$k^* = C_{[3]}-\alpha = C_{1\ 1}-\alpha = 16\ 55$ **(fifth step)** and the minimum penalty value is 14 55 **(sixth step)**

Another possible sequence of the groups of jobs could be F1 - F2 and the sequence of the jobs would be (1,1) - (1,2) - (2,1) - (2,2) respectively

For this case $k^* = C_{[3]} -\alpha = C_{2\ 1} = 9\ 55$ and the penalty value would be 24 55

We notice that applying our algorithm we achieved penalty value 14 55 < 24 55, which means that the schedule we obtain applying our algorithm is "better" because we obtain smaller penalty value (14 55 < 24 55)

5 3 Conclusions

In this chapter we presented an algorithm for scheduling groups of jobs on a single machine

Three Lemmas were presented and a numerical example was provided in order to illustrate the use of the results to determine the optimal solution to the due-date determination and sequencing problem

While it is fully appreciated that in practice penalty costs for earliness and tardiness are rarely the same, we imposed the restriction that weights $w_{i,j} = 1 \; \forall \; i,j$ were restricted to be 1

The reason is that the objective function that is used places emphasis on missing job due-dates Although it seems that this restriction has the disadvantage that it limits comparisons between the proposed algorithm and the main competitor, we can overcome this disadvantage by performing the "competing" algorithm considering a "hypothetical" case where the weight for each job is restricted to be 1

In order to evaluate the performance of this algorithm, the algorithm was coded in C++

The results that are obtained from this algorithm can be found in Appendix A

CHAPTER 6

AN ALGORITHM FOR THE DUE-DATE DETERMINATION AND SEQUENCING PROBLEM

6 0 Introduction

In this chapter we present an algorithm for the due-date determination and sequencing problem

This algorithm was developed in 1987 by Cheng [C87] and will be used in order to compare the results with the (JGA) algorithm

Because to our knowledge there is no published work that combines the features of family setup times with earliness / tardiness cost two features that are fundamental to many problems in practice we use this algorithm because its objective function is more relevant to our problem

6 1 Cheng's Algorithm

This Algorithm [C87] considers the problem of assigning due-dates and sequencing a given set of jobs on a single machine There will be penalties for completing jobs either ahead or behind their scheduled dates

The objective is to minimize a function of missing the job due-dates An algorithm is presented for determining the optimal due-dates and optimal job sequence simultaneously

Actually the objective is to determine the optimal constant flow allowance k^* and the optimal job sequence σ^* to minimize the weighted average of missed due-dates

Due date determination has been a popular research topic (for the single family case) and plentiful fruitful results have been obtained over the years

The popularity of scheduling research is due to the fact that the problem itself is theoretically challenging and the results are of practical usefulness This is because missing job due dates are entails such penalties as accumulating unnecessary stocks and/ or loss of production efficiency and customer goodwill

This algorithm considers an n-job, single machine scheduling problem with common due-dates

Let N denote the set of n independent jobs to be processed The jobs have the same starting times Job i requires t_i time units for processing and has a weighting factor w_i ($0<w_i<1$) and $\sum_{i \in N} w_i = 1, \forall i \in N$

The common due-date assignment method is employed to assign due-dates to jobs

- The job processing times t_i $\forall i \in N$ are known and deterministic
- The jobs are available for processing at the same time, i e $r_i = 0$ $\forall i \in N$
- There is a single machine available, which can only process the jobs one at a time
- Job splitting and preemption are not allowed

Let Π be the set of all possible job sequences and σ be an arbitrary sequence Let the subscript [i] denote the job in position i of σ Let $E_{[i]}$, $T_{[i]}$ and $C_{[i]}$ be the earliness, tardiness and completion time of the ith job in σ respectively

Whenever a job is not completed exactly on its due-date costs will be incurred, regardless of its being early or late, it is reasonable to minimize an objective function which is related to the average amount of missed due-dates

For this purpose we adopt, the weighted average of the absolute value of job lateness as the objective function to be minimized

While it is appreciated that, in practice, penalty costs for earhness and tardiness are not often the same, the use of the weighted average of absolute job lateness as the objective function places emphasis on missed job due-dates

For a given job sequence σ, let [i] denote the job in position i of σ In addition let $t_{[i]}$,$w_{[i]}$, $L_{[i]}$ and $d_{[i]}$ denote the the processing time, weighting factor, lateness, and due-date, respectively of job [i]

The objective function is expressed as

$$f(k,\sigma) = \sum_{i=1}^{n} w_{[i]}\left|L_{[i]}\right| = \sum_{i=1}^{n} w_{[i]}\left|C_{[i]} - d_{[i]}\right| = \sum_{i=1}^{n} w_{[i]}\left|C_{[i]} - k\right| \qquad (1)$$

CHENG'S ALGORITHM

Let n(X) denote the number of elements in a set X The algorithm systematically searches for the optimal solution as follows

STEP 1 Let r=1

STEP 2 Let $k=C_{[r]}$

STEP 3: Construct a set A where

$$A = \{S_A / S_A \subset N,\ n(S_A) = r,\ \sum_{\substack{i \in S_A - \{j\} \\ j \in S_A}} w_i \le 1/2 \ \ and \sum_{i \in S_A} w_i \ge 1/2\}$$

STEP 4 Construct a set B corresponding to A where

$$B = \{S_B / S_B \subset N,\ n(S_B) = (n-r),\ S_B = N - S_A,\ \forall S_A \in A\}$$

STEP 5 Arrange jobs in S_A, $\forall\ S_A \in A$, in nonincreasing order of $t_{[i]} / w_{[i]}$, $\forall\ i \in S_A$, to form a sequence σ_A and arrange jobs in S_B, $\forall\ S_B \in B$, in nondecreasing order $t_{[j]} / w_{[j]}$, $\forall\ j \in S_B$, to form a sequence σ_B Combine σ_A and σ_B to form a full sequence σ of n jobs, i e $\sigma = \sigma_A + \sigma_B$ Calculate the value of $f(k, \sigma)$ and record k, σ and $f(k, \sigma)$ for later evaluation

STEP 6 Let r = r+1 If r < n then go to Step2 else go to Step7

STEP 7 Identify $f^{*}(k, \sigma) = \min \{f(k, \sigma)\}$

Set $k^{*} = k$ of $f^{*}(k, \sigma)$ and $\sigma^{*} = \sigma$ of $f^{*}(k, \sigma)$

END OF THE ALGORITHM

6 2 Numerical example

To illustrate the operation of the algorithm, consider the following example There are five jobs with processing times and weighting factors given in Table 2 Table 3 shows the results obtained from performing the algorithm on the above given job-set

The algorithm has generated of 41 feasible sequences for consideration This feasible set of sequences is considerably smaller than the full set of all $5! = 120$ possible sequences

Thus, substantial saving in computations from employing the algorithm to search for the optimal solution

Table 2 Processing times and weighting factors of the numerical example

	Job 1	Job 2	Job 3	Job 4	Job 5
t_i	1	2	3	4	5
w_i	0.1	0.1	0.1	0.1	0.6
t_i/w_i	10	20	30	40	8+1/3

Table 3 Optimal solution

S_A	S_B	$\sigma=\sigma_A+\sigma_B$	$f(k,\sigma)$	k
5	1-2-3-4	5-1-2-3-4	2.0	5
5-1	2-3-4	1-5-2-3-4	2.1	6
5-2	1-3-4	2-5-1-3-4	1.8	7
5-3	1-2-4	3-5-1-2-4	1.6	8
5-4	1-2-3	4-5-1-2-3	1.5	9 *
5-1-2	3-4	2-1-5-3-4	2.1	8
5-1-3	2-4	3-1-5-2-4	1.9	9
5-1-4	2-3	4-1-5-2-3	1.8	10
5-2-1	3-4	2-1-5-3-4	2.1	8
5-2-3	1-4	3-2-5-1-4	1.8	11
5-2-4	1-3	4-2-5-1-3	1.7	12
5-3-1	2-4	3-1-5-2-4	1.9	9
5-3-2	1-4	3-2-5-1-4	1.8	11
5-3-4	1-2	4-3-5-1-2	1.7	13
5-4-1	2-3	4-1-5-2-3	1.8	10
5-4-2	1-3	4-2-5-1-3	1.7	12
5-4-3	1-2	4-3-5-1-2	1.7	13
5-1-2-3	4	3-2-1-5-4	2.3	11
5-1-2-4	3	4-2-1-5-3	2.2	12
5-1-3-2	4	4-2-1-5-3	2.3	11
5-1-3-4	2	4-3-1-5-2	2.2	13
5-1-4-2	3	4-2-1-5-3	2.2	12
5-1-4-3	2	4-3-1-5-2	2.2	13
5-2-1-3	4	3-2-1-5-4	2.3	11
5-2-1-4	3	4-2-1-5-3	2.2	12
5-2-3-1	4	4-2-1-5-3	2.3	11
5-2-3-4	1	4-3-2-5-1	2.3	14
5-2-4-1	3	4-2-1-5-3	2.2	12
5-2-4-3	1	4-3-2-5-1	2.3	14
5-3-1-2	4	3-2-1-5-4	2.3	11
5-3-1-4	2	4-3-1-5-2	2.2	13
5-3-2-1	4	3-2-1-5-4	2.3	11
5-3-2-4	1	4-3-2-5-1	2.3	14
5-3-4-1	2	4-3-1-5-2	2.2	13
5-3-4-2	1	4-3-2-5-1	2.3	14
5-4-1-2	3	4-2-1-5-3	2.2	12
5-4-1-3	2	4-3-1-5-2	2.2	13
5-4-2-1	3	4-2-1-5-3	2.2	12
5-4-2-3	1	4-3-2-5-1	2.3	14
5-4-3-1	2	4-3-1-5-2	2.2	13
5-4-3-2	1	4-3-2-5-1	2.3	14

It is clear that the minimum value of $f(k,\sigma)$ is $f(k,\sigma) = 1\ 5$ and thus $k^* = 1\ 5$ and

$\sigma^* = (4,5,1,2,3)$

6.3 Conclusions

In this chapter we described an algorithm for the due-date determination and sequencing problem. An example was presented to illustrate the performance of the algorithm to determine an optimal solution.

In order to evaluate the performance of this algorithm, the algorithm was coded in C++.

The results that are obtained from this algorithm can be found in Appendix B.

CHAPTER 7

PERFORMANCE AND EVALUATION

7 0 Introduction

In this Chapter we present the description of the data base of the test problems that we used in order to test our algorithm (JGA) in comparison with Cheng's algorithm In the last section of this chapter the conclusions after performing both algorithms, on the data base that we created are discussed and some ideas for further research are presented

The results we obtain from both algorithms are presented in Appendices A and B

7 1 Description of the database of test problems

In order to test both algorithms (JGA) and Cheng's algorithm we generated a database of test examples at random

The naming convention used for random test problems in the database is best explained by reference to some examples

N05G0Ex07 is the seventh example in the set of test problems with characteristics

- 5 jobs
- 0 families

N06G2N13N23Ex10 is the tenth example in the set of test problems with characteristics

- 6 jobs
- 2 families
- $n_1 = 3$
- $n_2 = 3$

The processing time for each job in a specific example is the same for both (JGA) and Cheng's algorithm

7.2 Conclusions

In this thesis an algorithm for scheduling groups of jobs on a single machine is presented

Our model differs from past models in the literature in that we consider an earliness / tardiness model with family set-up times We also incorporate a factor called completion time deviation allowance such that there will be no penalties if the completion time of job i is within the time interval $(d_i - \alpha, d_i + \alpha)$ $\forall\, i \in N$

To our knowledge, there is no published work on a family scheduling model with earliness and tardiness costs (in our model $w_{i,j} = 1 \quad \forall\, i,j$)

Our consideration of multiple families and a non regular performance measure, two features receiving increasing attention in the research community [BS90], [WB95], is motivated by real-world elements of practical scheduling problems

We have tested our algorithm on many examples, where the number of jobs varies from N=3 to N=50 and the number of families varies from G=0 to G=7 Unfortunately, because Cheng's algorithm is not performing at all for more than 7, jobs due to the enormous amount of computations that must be executed, we just present the results that we obtain from our algorithm for more than 7 jobs

Because Cheng's algorithm does not consider the feature of groups of jobs and family setup times, we perform the proposed algorithm (JGA) for the first 225 examples i e

N3G0Ex01-N3G0Ex45,

N4G0Ex01-N4G0Ex45,

N5G0Ex01-N5G0Ex45,

N6G0Ex01-N6G0Ex45,

N7G0Ex01-N7G0Ex45

assuming that the number of groups is zero (G=0) (i e we have a set of independent jobs) in order to compare the output from both algorithms under similar input data

The output we obtain from (JGA) algorithm and Cheng's algorithm is presented in appendices A and B

For example performing both algorithms for the problem instance N3G0Ex02 we obtain $\sigma^* = 1\text{-}3\text{-}2$, $k^*=6\ 65$, and the value of the objective function is 3 0,($\alpha = 0\ 35$ for this case) while from Cheng's algorithm we obtain $\sigma^* = 1\text{-}3\text{-}2$, $k^* = 6$, and the value of the objective function is 0 8

Although it seems that Cheng's algorithm is performing better than (JGA) algorithm, this is not valid, because the weights for jobs 1,2,3 are restricted to be 1 in the proposed algorithm, while the weights for jobs 1,2,3, in Cheng's algorithm are w_1=0 6, w_2=0 2 and w_3=0 2 (the sum of all weights must be $\sum_{i \in N} w_i = 1, \forall i \in N$)

Therefore because the objective function of Cheng's algorithm is

$$f(k,\sigma) = \sum_{i=1}^{n} w_{[i]} |L_{[i]}| = \sum_{i=1}^{n} w_{[i]} |C_{[i]} - d_{[i]}| = \sum_{i=1}^{n} w_{[i]} |C_{[i]} - k|$$

and $w_i < 1 \ \forall i \in \{1,2,3\}$ we obtain the value 0 8

Although it seems that this restriction (that weights $w_{i,j} = 1 \ \forall i,j$ are restricted to be 1) has the disadvantage that it limits comparisons between the proposed algorithm and the main competitor, we can overcome this disadvantage by performing the "competing" algorithm considering a "hypothetical" case where the weight for each job is restricted to be 1

We applied this "hypothetical" case for the following examples

N3G0Ex01-N3G0Ex45,

N4G0Ex01-N4G0Ex45,

N5G0Ex01-N5G0Ex45,

N6G0Ex01-N6G0Ex45,

N7G0Ex01-N7G0Ex45

For all these 225 examples we did not find an example where the (hypothetical) value for the objective function of Cheng's algorithm, is less than the value for the objective function of (JGA) algorithm

The (hypothetical) value for the objective function of Cheng's algorithm was either equal or greater than the value of the objective function of (JGA) algorithm for all 225 examples

Considering that fact, we can say that the results that we obtain from (JGA) algorithm are reasonably good

The main advantage of our algorithm is that it performs for many jobs while Cheng's algorithm can not perform for more than 7 jobs

For up to 7 jobs (JGA) algorithm produces results in considerably less time than Cheng's algorithm

We have performed our algorithm for up to 50 jobs and we observe that the CPU time was actually negligible and the results are reasonably good

In summary, the proposed algorithm appears to perform quite well when compared to Cheng's algorithm

7 3 Further Research

The basic features of the model we have studied represent a growth area in the scheduling literature and, consequently there are many opportunities for further research

The present problem can readily be generalized by introducing different penalties for earliness and tardiness as well as adding a penalty for assigning long due dates

Out model could also be generalized by not considering the GT assumption (i e the requirement of precisely f setups in the schedule one for each family, where f is the number of families)

Other significant generalizations to the model include

(a) Multiple machines (i e Groups of jobs can be scheduled on multiple machines that are placed together in a serial order)

(b) Parallel Machines (i e Groups of jobs can be scheduled on parallel machines)

(c) Dynamic job arrivals

Considering our model, a certain number of jobs arrive simultaneously to a system that is idle and is immediately available for work

A significant generalization to our model include Dynamic job arrivals Therefore jobs arrive intermittently at times that are predictable only in a statistical sense

(d) Job splitting and preemption

A significant generalization to our model could include the allowance of job splitting and preemption Therefore the processing of each job may be interrupted and resumed at a later time

APPENDIX A

OUTPUT FROM JGA ALGORITHM

	JGA			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N3G0Ex01	2-3-1	7.65	7	0.0
N3G0Ex02	1-3-2	6.65	3	0.0
N3G0Ex03	3-2-1	12.65	9	0.0
N3G0Ex04	3-2-1	11.65	11	0.0
N3G0Ex05	2-1-3	9.65	6	0.0
N3G0Ex06	3-1-2	5.65	3	0.0
N3G0Ex07	1-2-3	5.65	5	0.0
N3G0Ex08	3-1-2	5.65	5	0.0
N3G0Ex09	2-1-3	5.65	5	0.0
N3G0Ex10	3-1-2	7.65	4	0.0
N3G0Ex11	3-2-1	6.75	5	0.0
N3G0Ex12	3-2-1	9.75	9	0.0
N3G0Ex13	1-2-3	5.75	4	0.0
N3G0Ex14	3-2-1	8.75	5	0.0
N3G0Ex15	2-1-3	18.75	17	0.0
N3G0Ex16	2-1-3	7.75	6	0.0
N3G0Ex17	1-2-3	13.75	13	0.0
N3G0Ex18	3-2-1	14.75	13	0.0
N3G0Ex19	3-2-1	9.75	8	0.0
N3G0Ex20	1-2-3	14.75	11	0.0
N3G0Ex21	3-2-1	5.85	5	0.0
N3G0Ex22	3-1-2	8.85	8	0.0
N3G0Ex23	1-3-2	5.85	4	0.0
N3G0Ex24	3-1-2	4.55	3	0.0
N3G0Ex25	2-1-3	3.55	3	0.0
N3G0Ex26	3-2-1	5.55	3	0.0
N3G0Ex27	3-1-2	2.55	3	0.0
N3G0Ex28	3-2-1	2.55	3	0.0
N3G0Ex29	3-1-2	3.55	3	0.0
N3G0Ex30	1-3-2	3.55	3	0.0
N3G0Ex31	2-1-3	10.7	9	0.0
N3G0Ex32	2-3-1	10.7	9	0.0
N3G0Ex33	1-2-3	10.7	9	0.0
N3G0Ex34	2-1-3	6.7	4	0.0
N3G0Ex35	3-2-1	11.7	10	0.0
N3G0Ex36	3-2-1	5.7	6	0.0
N3G0Ex37	2-1-3	6.7	5	0.0
N3G0Ex38	1-2-3	7.8	6	0.0
N3G0Ex39	2-3-1	5.8	5	0.0
N3G0Ex40	1-2-3	5.8	4	0.0
N3G0Ex41	2-1-3	7.8	4	0.0
N3G0Ex42	1-3-2	5.8	4	0.0
N3G0Ex43	2-3-1	14.8	13	0.0
N3G0Ex44	3-2-1	8.8	8	0.0
N3G0Ex45	1-3-2	5.8	5	0.0

	JGA			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N4G0Ex01	2-4-1-3	4.55	4.55	0.0
N4G0Ex02	4-1-2-3	9.55	10.55	0.0
N4G0Ex03	2-3-4-1	7.55	6.55	0.0
N4G0Ex04	2-3-1-4	8.55	8.55	0.0
N4G0Ex05	1-2-4-3	8.55	8.55	0.0
N4G0Ex06	2-4-3-1	5.55	5.55	0.0
N4G0Ex07	4-2-1-3	6.7	6.7	0.0
N4G0Ex08	4-1-2-3	6.7	5.7	0.0
N4G0Ex09	4-3-1-2	8.7	9.7	0.0
N4G0Ex10	4-1-3-2	9.7	10.7	0.0
N4G0Ex11	3-4-2-1	4.7	4.7	0.0
N4G0Ex12	3-1-2-4	8.7	8.7	0.0
N4G0Ex13	4-1-2-3	6.7	6.7	0.0
N4G0Ex14	4-2-1-3	13.8	10.8	0.0
N4G0Ex15	2-3-1-4	16.8	15.8	0.0
N4G0Ex16	1-4-3-2	19.8	20.8	0.0
N4G0Ex17	4-2-1-3	15.8	12.8	0.0
N4G0Ex18	3-1-4-2	14.8	9.8	0.0
N4G0Ex19	1-3-4-2	12.8	10.8	0.0
N4G0Ex20	2-4-3-1	12.8	8.8	0.0
N4G0Ex21	4-2-1-3	16.9	15.9	0.0
N4G0Ex22	3-2-1-4	14.9	11.9	0.0
N4G0Ex23	4-1-2-3	16.9	14.9	0.0
N4G0Ex24	3-1-4-2	10.9	10.9	0.0
N4G0Ex25	4-3-2-1	8.9	7.9	0.0
N4G0Ex26	4-3-1-2	8.9	6.9	0.0
N4G0Ex27	4-3-1-2	15.9	13.9	0.0
N4G0Ex28	4-2-1-3	9.65	9.65	0.0
N4G0Ex29	3-4-1-2	14.65	14.65	0.0
N4G0Ex30	4-2-3-1	8.65	8.65	0.0
N4G0Ex31	3-1-2-4	16.65	16.65	0.0
N4G0Ex32	2-4-3-1	13.65	14.65	0.0
N4G0Ex33	2-3-4-1	10.65	12.65	0.0
N4G0Ex34	4-1-3-2	10.65	10.65	0.0
N4G0Ex35	1-2-4-3	18.75	22.75	0.0
N4G0Ex36	2-1-3-4	11.75	12.75	0.0
N4G0Ex37	1-3-2-4	16.75	18.75	0.0
N4G0Ex38	1-3-4-2	11.75	10.75	0.0
N4G0Ex39	2-3-1-4	9.75	9.75	0.0
N4G0Ex40	3-4-1-2	13.75	14.75	0.0
N4G0Ex41	1-4-3-2	12.75	14.75	0.0
N4G0Ex42	4-2-3-1	17.75	18.75	0.0
N4G0Ex43	4-1-3-2	14.75	16.75	0.0
N4G0Ex44	1-2-4-3	19.75	23.75	0.0
N4G0Ex45	1-3-4-2	13.75	14.75	0.0

	JGA			
	σ^*	K^*	$f(k^*,\sigma^*)$	CPU time
N5G0Ex01	4-2-1-5-3	13.55	15	0.0
N5G0Ex02	5-1-4-3-2-	9.55	13	0.0
N5G0Ex03	4-2-5-1-3	12.55	15	0.0
N5G0Ex04	3-2-1-4-5	10.55	15	0.0
N5G0Ex05	1-2-4-3-5	9.55	14	0.0
N5G0Ex06	5-3-2-1-4	14.55	16	0.0
N5G0Ex07	5-3-1-2-4	8.7	13	0.0
N5G0Ex08	5-2-4-1-3	14.7	15	0.0
N5G0Ex09	5-1-4-2-3	12.7	16	0.0
N5G0Ex10	2-5-1-3-4	11.7	21	0.0
N5G0Ex11	5-2-1-3-4	15.7	20	0.0
N5G0Ex12	4-5-2-1-3	11.7	16	0.0
N5G0Ex13	5-2-1-3-4	9.8	14	0.0
N5G0Ex14	1-5-3-2-4	6.8	11	0.0
N5G0Ex15	5-2-1-3-4	13.8	18	0.0
N5G0Ex16	5-2-4-1-3	13.8	17	0.0
N5G0Ex17	4-2-1-3-5	9.8	13	0.0
N5G0Ex18	5-4-2-3-1	11.8	16	0.0
N5G0Ex19	4-5-1-2-3	14.9	25	0.0
N5G0Ex20	5-2-4-1-3	13.9	20	0.0
N5G0Ex21	5-3-4-2-1	13.9	16	0.0
N5G0Ex22	4-2-3-1-5	10.9	15	0.0
N5G0Ex23	5-1-3-2-4	12.9	15	0.0
N5G0Ex24	5-3-2-1-4	12.9	16	0.0
N5G0Ex25	4-1-3-5-2	8.85	13	0.0
N5G0Ex26	4-5-3-1-2	9.85	13	0.0
N5G0Ex27	5-1-3-2-4	10.85	15	0.0
N5G0Ex28	5-2-4-1-3	10.85	15	0.0
N5G0Ex29	2-4-1-3-5	19.85	32	0.0
N5G0Ex30	1-3-5-2-4	19.85	34	0.0
N5G0Ex31	1-2-3-4-5	9.65	13	0.0
N5G0Ex32	5-1-2-4-3	10.65	15	0.0
N5G0Ex33	3-5-4-1-2	24.65	41	0.0
N5G0Ex34	4-3-2-1-5	15.65	22	0.0
N5G0Ex35	4-2-1-5-3	24.65	43	0.0
N5G0Ex36	2-4-5-1-3	17.65	31	0.0
N5G0Ex37	5-3-4-1-2	11.75	19	0.0
N5G0Ex38	5-1-2-3-4	18.75	32	0.0
N5G0Ex39	5-1-2-3-4	19.75	35	0.0
N5G0Ex40	4-1-3-2-5	14.75	23	0.0
N5G0Ex41	3-4-1-5-2	18.75	32	0.0
N5G0Ex42	4-1-2-5-3	19.75	32	0.0
N5G0Ex43	4-5-2-1-3	15.9	25	0.0
N5G0Ex44	3-2-4-1-5	13.9	23	0.0
N5G0Ex45	3-5-4-1-2	14.9	23	0.0

	JGA			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N6G0Ex01	4-6-5-3-2-1	10.55	19.55	0.0
N6G0Ex02	5-4-3-2-6-1	14.55	21.55	0.0
N6G0Ex03	6-4-2-1-3-5	12.55	21.55	0.0
N6G0Ex04	1-2-3-4-5-6	14.55	20.55	0.0
N6G0Ex05	6-3-2-5-1-4	17.55	25.55	0.0
N6G0Ex06	3-6-4-5-1-2	12.6	21.6	0.0
N6G0Ex07	5-2-1-3-6-4	14.6	21.6	0.0
N6G0Ex08	4-3-6-5-1-2	12.6	27.6	0.0
N6G0Ex09	6-1-3-5-2-4	16.6	23.6	0.0
N6G0Ex10	6-1-3-5-2-4	14.6	21.65	0.0
N6G0Ex11	5-2-4-3-1-6	12.65	21.65	0.0
N6G0Ex12	5-1-2-3-4-6	12.65	21.65	0.0
N6G0Ex13	5-1-6-4-2-3	13.65	21.65	0.0
N6G0Ex14	6-5-1-2-3-4	17.65	27.65	0.0
N6G0Ex15	6-1-5-3-2-4	12.65	21.65	0.0
N6G0Ex16	6-2-5-3-1-4	13.75	22.75	0.0
N6G0Ex17	6-1-4-3-5-2	14.75	21.75	0.0
N6G0Ex18	4-6-3-5-2-1	12.75	21.75	0.0
N6G0Ex19	6-5-2-3-1-4	13.75	21.75	0.0
N6G0Ex20	6-1-3-4-2-5	12.75	21.75	0.0
N6G0Ex21	6-4-2-3-1-5	17.8	28.8	0.0
N6G0Ex22	6-2-5-4-1-3	19.8	34.8	0.0
N6G0Ex23	2-1-5-4-3-6	17.8	30.8	0.0
N6G0Ex24	6-5-4-3-2-1	11.8	19.8	0.0
N6G0Ex25	6-1-2-5-3-4	14.8	23.8	0.0
N6G0Ex26	6-3-4-5-2-1	12.7	21.7	0.0
N6G0Ex27	6-5-2-4-1-3	18.7	33.7	0.0
N6G0Ex28	6-3-2-4-1-5	22.7	33.7	0.0
N6G0Ex29	5-6-1-3-2-4	24.7	35.7	0.0
N6G0Ex30	6-4-1-2-3-5	20.7	34.7	0.0
N6G0Ex31	6-5-3-2-4-1	16.55	30.55	0.0
N6G0Ex32	6-3-2-5-4-1	17.55	31.55	0.0
N6G0Ex33	3-5-4-1-2-6	15.55	27.55	0.0
N6G0Ex34	2-1-4-6-5-3	18.55	34.55	0.0
N6G0Ex35	6-3-4-2-1-5	12.55	21.55	0.0
N6G0Ex36	2-3-6-5-4-1	12.7	21.7	0.0
N6G0Ex37	2-4-3-5-6-1	19.7	36.7	0.0
N6G0Ex38	5-3-4-2-1-6	15.7	26.7	0.0
N6G0Ex39	6-5-2-4-1-3	15.7	23.7	0.0
N6G0Ex40	5-4-3-1-6-2	19.7	37.7	0.0
N6G0Ex41	5-6-3-4-2-1	16.8	25.8	0.0
N6G0Ex42	4-1-5-2-3-6	10.8	16.8	0.0
N6G0Ex43	6-4-5-1-2-3	18.8	32.8	0.0
N6G0Ex44	6-3-5-1-4-2	16.8	30.8	0.0
N6G0Ex45	5-6-2-1-3-4	13.8	22.8	0.0

	JGA			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N7G0Ex01	6-4-5-1-7-2-3	12.55	25	0.0
N7G0Ex02	5-6-4-7-1-2-3	13.55	32	0.0
N7G0Ex03	7-4-5-2-3-1-6	16.55	34	0.0
N7G0Ex04	7-6-5-4-1-3-2	15.55	34	0.0
N7G0Ex05	7-3-5-4-1-2-6	17.55	35	0.0
N7G0Ex06	6-4-3-2-1-7-5	20.6	45	0.0
N7G0Ex07	7-3-1-2-4-5-6	18.6	42	0.0
N7G0Ex08	3-2-1-5-7-6-4	18.6	42	0.0
N7G0Ex09	7-5-3-2-1-4-6	18.6	37	0.0
N7G0Ex10	4-3-5-2-7-1-6	15.6	36	0.0
N7G0Ex11	6-5-1-3-2-4-7	19.65	46	0.0
N7G0Ex12	2-5-1-4-3-6-7	14.65	33	0.0
N7G0Ex13	6-7-4-2-1-3-5	18.65	44	0.0
N7G0Ex14	6-3-4-1-2-7-5	17.65	42	0.0
N7G0Ex15	5-2-3-7-6-1-4	15.65	34	0.0
N7G0Ex16	5-4-1-6-3-7-2	13.7	30	0.0
N7G0Ex17	7-4-2-5-1-3-6	19.7	41	0.0
N7G0Ex18	5-4-3-2-7-1-6	16.7	35	0.0
N7G0Ex19	7-4-5-6-1-2-3	16.7	34	0.0
N7G0Ex20	5-2-1-6-3-7-4	17.7	42	0.0
N7G0Ex21	7-6-3-2-1-4-5	13.75	32	0.0
N7G0Ex22	6-7-4-5-1-3-2	13.75	32	0.0
N7G0Ex23	4-5-3-1-7-2-6	17.75	37	0.0
N7G0Ex24	6-5-3-1-2-7-4	15.75	37	0.0
N7G0Ex25	6-5-3-1-4-2-7	18.75	43	0.0
N7G0Ex26	2-7-5-6-4-1-3	13.8	32	0.0
N7G0Ex27	5-4-7-6-2-3-1	14.8	30	0.0
N7G0Ex28	4-6-5-2-3-7-1	15.8	34	0.0
N7G0Ex29	3-5-2-7-1-4-6	16.8	34	0.0
N7G0Ex30	6-5-2-4-3-1-7	18.8	43	0.0
N7G0Ex31	7-2-1-6-3-5-4	16.85	34	0.0
N7G0Ex32	7-6-5-4-1-2-3	31.85	82	0.0
N7G0Ex33	3-2-6-7-1-5-4	28.85	66	0.0
N7G0Ex34	7-4-2-3-5-1-6	33.85	76	0.0
N7G0Ex35	6-3-2-5-1-4-7	27.85	64	0.0
N7G0Ex36	5-1-7-6-2-3-4	21.9	48	0.0
N7G0Ex37	4-3-7-5-2-6-1	20.9	50	0.0
N7G0Ex38	4-6-5-7-2-3-1	21.9	47	0.0
N7G0Ex39	4-3-6-1-2-5-7	28.9	73	0.0
N7G0Ex40	4-3-2-7-1-6-5	26.9	56	0.0
N7G0Ex41	7-4-6-5-2-3-1	15.9	34	0.0
N7G0Ex42	7-1-4-5-6-2-3	19.9	46	0.0
N7G0Ex43	1-6-4-5-7-3-2	17.9	34	0.0
N7G0Ex44	1-2-7-6-3-4-5	16.9	34	0.0
N7G0Ex45	1-4-2-5-3-7-6	15.9	34	0.0

	JGA			
	σ^*	K^*	$f(k^*,\sigma^*)$	CPU time
N4G2N$_1$1N$_2$3 Ex01	F1-F2	12 55	15 55	0 1
N4G2N$_1$2N$_2$2 Ex02	F2-F1	14 55	17 55	0 1
N4G2N$_1$3N$_2$1 Ex03	F2-F1	11 55	14 55	0 1
N5G2N$_1$1N$_2$4 Ex01	F1-F2	12 55	22	0 1
N5G2N$_1$2N$_2$3 Ex02	F1-F2	12 55	22	0 1
N5G2N$_1$3N$_2$2 Ex03	F1-F2	12 55	22	0 1
N5G2N$_1$4N$_2$1 Ex04	F1-F2	12 55	22	0 1
N6G2N$_1$1N$_2$5 Ex01	F1-F2	16 7	28 7	0 1
N6G2N$_1$2N$_2$4 Ex02	F1-F2	16 7	28 7	0 1
N6G2N$_1$3N$_2$3 Ex03	F1-F2	16 7	28 7	0 1
N6G2N$_1$4N$_2$2 Ex04	F1-F2	16 7	28 7	0 1
N6G2N$_1$5N$_2$1 Ex05	F1-F2	16 7	28 7	0 1
N7G2N$_1$6N$_2$1 Ex01	F1-F2	12 65	35	0 1
N7G2N$_1$5N$_2$2 Ex02	F1-F2	12 65	35	0 1
N7G2N$_1$4N$_2$3 Ex03	F1-F2	12 65	35	0 1
N7G2N$_1$3N$_2$4 Ex04	F1-F2	12 65	35	0 1
N7G2N$_1$2N$_2$5 Ex05	F2-F1	12 65	22	0 1
N7G2N$_1$1N$_2$6 Ex06	F2-F1	12 65	29	0 1
N8G2N$_1$7N$_2$1 Ex01	F2-F1	16 75	47 75	0 1
N8G2N$_1$6N$_2$2 Ex02	F2-F1	14 75	51 75	0 1
N8G2N$_1$5N$_2$3 Ex03	F1-F2	14 75	43 75	0 1
N8G2N$_1$4N$_2$4 Ex04	F1-F2	14 75	43 75	0 1
N8G2N$_1$3N$_2$5 Ex05	F1-F2	14 75	43 75	0 1
N8G2N$_1$2N$_2$6 Ex06	F2-F1	14 75	35 75	0 1
N8G2N$_1$1N$_2$7 Ex07	F2-F1	13 75	39 75	0 1
N9G2N$_1$8N$_2$1 Ex01	F2-F1	19 85	57	0 1
N9G2N$_1$7N$_2$2 Ex02	F2-F1	19 85	64	0 1
N9G2N$_1$6N$_2$3 Ex03	F2-F1	15 85	69	0 1
N9G2N$_1$5N$_2$4 Ex04	F2-F1	15 85	71	0 1
N9G2N$_1$4N$_2$5 Ex05	F1-F2	14 85	58	0 1
N9G2N$_1$3N$_2$6 Ex06	F2-F1	12 85	68	0 1
N9G2N$_1$2N$_2$7 Ex07	F2-F1	14 85	61	0 1
N9G2N$_1$1N$_2$8 Ex08	F2-F1	13 85	60	0 1
N10G2N$_1$9N$_2$1 Ex01	F1-F2	19 8	79 8	0 1
N10G2N$_1$8N$_2$2 Ex02	F1-F2	19 8	79 8	0 1
N10G2N$_1$7N$_2$3 Ex03	F2-F1	27 8	86 8	0 1
N10G2N$_1$6N$_2$4 Ex04	F2-F1	24 8	95 8	0 1
N10G2N$_1$5N$_2$5 Ex05	F1-F2	19 8	79 8	0 1
N10G2N$_1$4N$_2$6 Ex06	F1-F2	19 8	79 8	0 1
N10G2N$_1$3N$_2$7 Ex07	F1-F2	19 8	79 8	0 1
N10G2N$_1$2N$_2$8 Ex08	F1-F2	19 8	79 8	0 1
N10G2N$_1$1N$_2$9 Ex09	F1-F2	19 8	79 8	0 1

	JGA			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N11G3$N_1$1$N_2$1$N_3$9 Ex01	F3-F2-F1	18 6	107	0 2
N11G3$N_1$2$N_2$1$N_3$8 Ex02	F2-F1-F3	15 6	89	0 2
N11G3$N_1$2$N_2$2$N_3$7 Ex03	F2-F1-F3	15 6	84	0 2
N11G3$N_1$2$N_2$3$N_3$6 Ex04	F2-F1-F3	15 6	83	0 2
N11G3$N_1$3$N_2$3$N_3$5 Ex05	F3-F2-F1	24 6	93	0 2
N11G3$N_1$3$N_2$4$N_3$4 Ex06	F3-F2-F1	24 6	79	0 2
N11G3$N_1$4$N_2$4$N_3$3 Ex07	F3-F2-F1	27 6	94	0 2
N11G3$N_1$4$N_2$3$N_3$4 Ex08	F3-F2-F1	21 6	76	0 2
N11G3$N_1$4$N_2$5$N_3$2 Ex09	F3-F2-F1	16 6	100	0 2
N11G3$N_1$4$N_2$2$N_3$5 Ex10	F3-F2-F1	22 6	87	0 2
N11G3$N_1$2$N_2$4$N_3$5 Ex11	F3-F2-F1	22 6	101	0 2
N12G3$N_1$1$N_2$1$N_3$10 Ex01	F3-F2-F1	25 6	127 6	0 2
N12G3$N_1$2$N_2$2$N_3$8 Ex02	F2-F1-F3	17 6	106 6	0 2
N12G3$N_1$2$N_2$3$N_3$7 Ex03	F2-F1-F3	17 6	105 6	0 2
N12G3$N_1$3$N_2$3$N_3$6 Ex04	F3-F2-F1	26 6	106 6	0 2
N12G3$N_1$4$N_2$3$N_3$5 Ex05	F3-F2-F1	23 6	86 6	0 2
N12G3$N_1$4$N_2$4$N_3$4 Ex06	F3-F2-F1	21 6	88 6	0 2
N12G3$N_1$5$N_2$4$N_3$3 Ex07	F2-F1-F3	22 6	130 6	0 2
N12G3$N_1$4$N_2$5$N_3$3 Ex08	F3-F2-F1	18 6	112 6	0 2
N12G3$N_1$6$N_2$3$N_3$3 Ex09	F2-F1-F3	25 6	118 6	0 2
N12G3$N_1$3$N_2$6$N_3$3 Ex10	F3-F1-F2	22 6	105 6	0 2
N12G3$N_1$2$N_2$6$N_3$4 Ex11	F3-F1-F2	25 6	104 6	0 2
N12G3$N_1$2$N_2$5$N_3$5 Ex12	F3-F1-F2	23 6	105 6	0 2
N12G3$N_1$3$N_2$5$N_3$4 Ex13	F3-F2-F1	23 6	88 6	0 2
N12G3$N_1$5$N_2$5$N_3$2 Ex14	F2-F3-F1	23 6	119 6	0 2
N13G3$N_1$1$N_2$1$N_3$11 Ex01	F3-F2-F1	25 9	153	0 2
N13G3$N_1$2$N_2$2$N_3$9 Ex02	F2-F1-F3	17 9	133	0 2
N13G3$N_1$3$N_2$3$N_3$7 Ex03	F3-F2-F1	27 9	137	0 2
N13G3$N_1$3$N_2$4$N_3$6 Ex04	F3-F2-F1	29 9	122	0 2
N13G3$N_1$3$N_2$5$N_3$5 Ex05	F3-F2-F1	27 9	115	0 2
N13G3$N_1$4$N_2$4$N_3$5 Ex06	F3-F2-F1	24 9	112	0 2
N13G3$N_1$5$N_2$4$N_3$4 Ex07	F3-F1-F2	23 9	130	0 2
N13G3$N_1$4$N_2$6$N_3$3 Ex08	F2-F1-F3	22 9	152	0 2
N13G3$N_1$5$N_2$5$N_3$3 Ex09	F2-F1-F3	21 9	149	0 2
N13G3$N_1$4$N_2$5$N_3$4 Ex10	F3-F2-F1	19 9	133	0 2
N13G3$N_1$5$N_2$6$N_3$2 Ex11	F2-F1-F3	23 9	145	0 2
N13G3$N_1$6$N_2$5$N_3$2 Ex12	F2-F1-F3	23 9	137	0 2
N13G3$N_1$7$N_2$2$N_3$4 Ex13	F2-F1-F3	25 9	124	0 2
N13G3$N_1$3$N_2$7$N_3$3 Ex14	F2-F1-F3	24 9	150	0 2
N13G3$N_1$4$N_2$6$N_3$3 Ex15	F2-F1-F3	22 9	152	0 2

	JGA			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N14G3$N_1$1$N_2$1$N_3$12 Ex01	F3-F2-F1	30 9	170 9	0 4
N14G3$N_1$2$N_2$2$N_3$10 Ex02	F2-F1-F3	21 9	159 9	0 3
N14G3$N_1$3$N_2$5$N_3$6 Ex03	F3-F2-F1	28 9	125 9	0 3
N14G3$N_1$3$N_2$6$N_3$5 Ex04	F2-F1-F3	22 9	169 9	0 3
N14G3$N_1$4$N_2$5$N_3$5 Ex05	F1-F3-F2	27 9	147 9	0 3
N14G3$N_1$4$N_2$4$N_3$6 Ex06	F3-F2-F1	25 9	119 9	0 3
N14G3$N_1$5$N_2$4$N_3$5 Ex07	F2-F1-F3	26 9	173 9	0 3
N14G3$N_1$5$N_2$5$N_3$4 Ex08	F2-F3-F1	28 9	160 9	0 3
N14G3$N_1$6$N_2$4$N_3$4 Ex09	F2-F1-F3	30 9	155 9	0 3
N14G3$N_1$7$N_2$3$N_3$4 Ex10	F2-F1-F3	30 9	139 9	0 3
N14G3$N_1$5$N_2$6$N_3$3 Ex11	F2-F3-F1	28 9	160 9	0 3
N14G3$N_1$7$N_2$5$N_3$2 Ex12	F2-F1-F3	29 9	146 9	0 3
N14G3$N_1$6$N_2$6$N_3$2 Ex13	F2-F1-F3	25 9	151 9	0 3

	JGA (α = 0 3)		
	k^*	$f(k^*,\sigma^*)$	CPU time
N20G4$N_1$5$N_2$5$N_3$5$N_4$5Ex01	44 7	389 7	0 5
N20G4$N_1$10$N_2$3$N_3$3$N_4$4Ex02	48 7	375 7	0 5
N20G4$N_1$9$N_2$3$N_3$5$N_4$3Ex03	44 7	379 7	0 5
N20G4$N_1$2$N_2$8$N_3$5$N_4$5Ex04	48 7	388 7	0 5
N20G4$N_1$3$N_2$7$N_3$5$N_4$5Ex05	47 7	375 7	0 5
N20G4$N_1$5$N_2$8$N_3$2$N_4$5Ex06	46 7	397 7	0 5
N20G4$N_1$2$N_2$3$N_3$10$N_4$5Ex07	47 7	427 7	0 5
N20G4$N_1$7$N_2$3$N_3$5$N_4$5Ex08	46 7	366 7	0 5
N20G4$N_1$8$N_2$2$N_3$5$N_4$5Ex09	44 7	371 7	0 5
N20G4$N_1$4$N_2$6$N_3$5$N_4$5Ex10	45 7	372 7	0 5

	σ^*
N20G4$N_1$5$N_2$5$N_3$5$N_4$5Ex01	F3-F4-F2-F1
N20G4$N_1$10$N_2$3$N_3$3$N_4$4Ex02	F3-F4-F1-F2
N20G4$N_1$9$N_2$3$N_3$5$N_4$3Ex03	F2-F4-F1-F3
N20G4$N_1$2$N_2$8$N_3$5$N_4$5Ex04	F3-F4-F2-F1
N20G4$N_1$3$N_2$7$N_3$5$N_4$5Ex05	F3-F4-F2-F1
N20G4$N_1$5$N_2$8$N_3$2$N_4$5Ex06	F3-F4-F2-F1
N20G4$N_1$2$N_2$3$N_3$10$N_4$5Ex07	F3-F4-F2-F1
N20G4$N_1$7$N_2$3$N_3$5$N_4$5Ex08	F2-F4-F1-F3
N20G4$N_1$8$N_2$2$N_3$5$N_4$5Ex09	F3-F1-F2-F4
N20G4$N_1$4$N_2$6$N_3$5$N_4$5Ex10	F3-F4-F2-F1

	JGA (α = 0.2)		
	k^*	$f(k^*,\sigma^*)$	CPU time
N30G5$N_1$5$N_2$5$N_3$5$N_4$10$N_5$5Ex01	63 8	829 8	0 7
N30G5$N_1$10$N_2$5$N_3$4$N_4$5$N_5$6Ex02	64 8	834 8	0 7
N30G5$N_1$5$N_2$6$N_3$6$N_4$6$N_5$7Ex03	61 8	883 8	0 7
N30G5$N_1$3$N_2$6$N_3$5$N_4$7$N_5$9Ex04	60 8	824 8	0 7
N30G5$N_1$6$N_2$5$N_3$4$N_4$8$N_5$7Ex05	65 8	805 8	0 7
N30G5$N_1$10$N_2$5$N_3$4$N_4$1$N_5$10Ex06	62 8	827 8	0 7
N30G5$N_1$5$N_2$6$N_3$6$N_4$7$N_5$6Ex07	59 8	825 8	0 7
N30G5$N_1$6$N_2$3$N_3$5$N_4$7$N_5$9Ex08	60 8	793 8	0 7
N30G5$N_1$3$N_2$7$N_3$5$N_4$10$N_5$5Ex09	66 8	906 8	0 7
N30G5$N_1$6$N_2$3$N_3$9$N_4$2$N_5$10Ex10	65 8	783 8	0 7

	σ^*
N30G5$N_1$5$N_2$5$N_3$5$N_4$10$N_5$5Ex01	F3-F2-F4-F5-F1
N30G5$N_1$10$N_2$5$N_3$4$N_4$5$N_5$6Ex02	F2-F3-F5-F1-F4
N30G5$N_1$5$N_2$6$N_3$6$N_4$6$N_5$7Ex03	F3-F1-F5-F4-F2
N30G5$N_1$3$N_2$6$N_3$5$N_4$7$N_5$9Ex04	F1-F4-F2-F5-F3
N30G5$N_1$6$N_2$5$N_3$4$N_4$8$N_5$7Ex05	F3-F4-F1-F5-F2
N30G5$N_1$10$N_2$5$N_3$4$N_4$1$N_5$10Ex06	F2-F3-F5-F1-F4
N30G5$N_1$5$N_2$6$N_3$6$N_4$7$N_5$6Ex07	F3-F2-F5-F1-F4
N30G5$N_1$6$N_2$3$N_3$5$N_4$7$N_5$9Ex08	F2-F4-F1-F5-F3
N30G5$N_1$3$N_2$7$N_3$5$N_4$10$N_5$5Ex09	F3-F5-F2-F4-F1
N30G5$N_1$6$N_2$3$N_3$9$N_4$2$N_5$10Ex10	F4-F3-F1-F5-F2

	JGA ($\alpha = 0.35$)		
	k^*	$f(k^*,\sigma^*)$	CPU time
$N40G6N_15N_25N_35N_45N_510N_610Ex01$	80 65	1439 65	1
$N40G6N_13N_24N_38N_410N_58N_67Ex02$	88 65	1454 65	1
$N40G6N_16N_26N_36N_46N_58N_68Ex03$	81 65	1431 65	1
$N40G6N_17N_27N_37N_45N_55N_69Ex04$	85 65	1407 65	1
$N40G6N_15N_25N_35N_410N_55N_610Ex05$	82 65	1450 65	1
$N40G6N_18N_23N_38N_410N_58N_67Ex06$	83 65	1518 65	1
$N40G6N_16N_26N_36N_48N_56N_68Ex07$	85 65	1447 65	1
$N40G6N_17N_27N_35N_47N_55N_69Ex08$	90 65	1475 65	1
$N40G6N_13N_29N_36N_48N_56N_68Ex09$	81 65	1519 65	1
$N40G6N_12N_212N_35N_47N_55N_69Ex10$	81 65	1489 65	1

	σ^*
$N40G6N_15N_25N_35N_45N_510N_610Ex01$	F3-F1-F2-F5-F4-F6
$N40G6N_13N_24N_38N_410N_58N_67Ex02$	F3-F6-F4-F2-F5-F1
$N40G6N_16N_26N_36N_46N_58N_68Ex03$	F2-F6-F1-F5-F3-F4
$N40G6N_17N_27N_37N_45N_55N_69Ex04$	F2-F3-F4-F1-F5-F6
$N40G6N_15N_25N_35N_410N_55N_610Ex05$	F3-F1-F5-F4-F2-F6
$N40G6N_18N_23N_38N_410N_58N_67Ex06$	F3-F5-F6-F4-F1-F2
$N40G6N_16N_26N_36N_48N_56N_68Ex07$	F2-F3-F4-F1-F5-F6
$N40G6N_17N_27N_35N_47N_55N_69Ex08$	F2-F6-F4-F1-F5-F3-
$N40G6N_13N_29N_36N_48N_56N_68Ex09$	F1-F2-F5-F4-F3-F6
$N40G6N_12N_212N_35N_47N_55N_69Ex10$	F3-F6-F5-F4-F2-F1

		JGA ($\alpha = 0.3$)	
	k^*	$f(k^*,\sigma^*)$	CPU time
$N50G7N_{1}10N_{2}5N_{3}5N_{4}5N_{5}10N_{6}10N_{7}5$ Ex01	97 7	2246 7	1 1
$N50G7N_{1}10N_{2}5N_{3}5N_{4}10N_{5}5N_{6}10N_{7}5$ Ex02	96 7	2243 7	1 1
$N50G7N_{1}10N_{2}5N_{3}5N_{4}5N_{5}10N_{6}5N_{7}10$ Ex03	99 7	2253 7	1 1
$N50G7N_{1}5N_{2}10N_{3}5N_{4}5N_{5}10N_{6}10N_{7}5$ Ex04	99 7	2274 7	1 1
$N50G7N_{1}10N_{2}5N_{3}5N_{4}7N_{5}3N_{6}10N_{7}10$ Ex05	102 7	2285 7	1 1
$N50G7N_{1}10N_{2}5N_{3}5N_{4}5N_{5}10N_{6}5N_{7}10$ Ex06	100 7	2254 7	1 1
$N50G7N_{1}10N_{2}3N_{3}2N_{4}10N_{5}10N_{6}10N_{7}5$ Ex07	102 7	2273 7	1 1
$N50G7N_{1}10N_{2}5N_{3}5N_{4}8N_{5}2N_{6}10N_{7}10$ Ex08	102 7	2276 7	1 1
$N50G7N_{1}10N_{2}5N_{3}15N_{4}5N_{5}10N_{6}5N_{7}5$ Ex09	99 7	2233 7	1 1
$N50G7N_{1}10N_{2}5N_{3}5N_{4}10N_{5}5N_{6}10N_{7}5$ Ex10	96 7	2243 7	1 1

	σ^*
$N50G7N_{1}10N_{2}5N_{3}5N_{4}5N_{5}10N_{6}10N_{7}5$ Ex01	F2-F3-F5-F7-F4-F1-F6
$N50G7N_{1}10N_{2}5N_{3}5N_{4}10N_{5}5N_{6}10N_{7}5$ Ex02	F2-F3-F5-F7-F4-F1-F6
$N50G7N_{1}10N_{2}5N_{3}5N_{4}5N_{5}10N_{6}5N_{7}10$ Ex03	F2-F3-F5-F4-F7-F1-F6
$N50G7N_{1}5N_{2}10N_{3}5N_{4}5N_{5}10N_{6}10N_{7}5$ Ex04	F2-F1-F5-F7-F4-F3-F6
$N50G7N_{1}10N_{2}5N_{3}5N_{4}7N_{5}3N_{6}10N_{7}10$ Ex05	F2-F6-F1-F4-F7-F4-F5
$N50G7N_{1}10N_{2}10N_{3}5N_{4}5N_{5}5N_{6}5N_{7}10$ Ex06	F2-F1-F4-F3-F7-F5-F6
$N50G7N_{1}10N_{2}3N_{3}2N_{4}10N_{5}10N_{6}10N_{7}5$Ex07	F3-F6-F5-F7-F4-F1-F2-
$N50G7N_{1}10N_{2}5N_{3}5N_{4}8N_{5}2N_{6}10N_{7}10$ Ex08	F2-F6-F1-F4-F7-F3-F5
$N50G7N_{1}10N_{2}5N_{3}15N_{4}5N_{5}5N_{6}5N_{7}5$ Ex09	F2-F6-F3-F7-F4-F1-F5
$N50G7N_{1}10N_{2}5N_{3}5N_{4}10N_{5}5N_{6}10N_{7}5$ Ex10	F2-F3-F5-F7-F4-F1-F6

APPENDIX B

OUTPUT FROM CHENG'S ALGORITHM

	CHENG'S ALGORITHM			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N3G0Ex01	2-3-1	8	1.7	0.1
N3G0Ex02	1-3-2	6	0.8	0.1
N3G0Ex03	1-3-2	14	1.3	0.1
N3G0Ex04	1-3-2	13	2.9	0.1
N3G0Ex05	2-1-3	10	1.4	0.1
N3G0Ex06	3-1-2	6	0.7	0.1
N3G0Ex07	1-3-2	7	1.1	0.1
N3G0Ex08	2-1-3	5	0.7	0.1
N3G0Ex09	2-3-1	7	1.2	0.1
N3G0Ex10	3-1-2	8	0.4	0.1
N3G0Ex11	3-1-2	8	1.1	0.1
N3G0Ex12	1-2-3	9	2.2	0.1
N3G0Ex13	1-3-2	8	0.8	0.1
N3G0Ex14	3-1-2	10	0.7	0.1
N3G0Ex15	2-1-3	19	1.7	0.1
N3G0Ex16	2-1-3	8	1.2	0.1
N3G0Ex17	1-2-3	14	2.3	0.1
N3G0Ex18	3-2-1	15	1.9	0.1
N3G0Ex19	1-3-2	12	1.3	0.1
N3G0Ex20	1-2-3	15	1.1	0.1
N3G0Ex21	1-2-3	5	0.8	0.1
N3G0Ex22	3-1-2	9	1	0.1
N3G0Ex23	1-3-2	5	0.5	0.1
N3G0Ex24	3-1-2	4	0.8	0.1
N3G0Ex25	2-3-1	5	0.8	0.1
N3G0Ex26	3-1-2	7	0.7	0.1
N3G0Ex27	3-1-2	3	0.6	0.1
N3G0Ex28	3-1-2	4	0.7	0.1
N3G0Ex29	3-1-2	4	0.5	0.1
N3G0Ex30	3-1-2	4	1.1	0.1
N3G0Ex31	2-1-3	11	2.1	0.1
N3G0Ex32	3-2-1	11	2.7	0.1
N3G0Ex33	3-1-2	12	1.9	0.1
N3G0Ex34	2-3-1	9	0.8	0.1
N3G0Ex35	3-1-2	14	2.2	0.1
N3G0Ex36	3-1-2	8	1.2	0.1
N3G0Ex37	2-3-1	8	1.3	0.1
N3G0Ex38	3-1-2	10	2.2	0.1
N3G0Ex39	2-1-3	9	0.7	0.1
N3G0Ex40	1-2-3	6	0.7	0.1
N3G0Ex41	2-1-3	8	1.0	0.1
N3G0Ex42	1-3-2	5	1.3	0.1
N3G0Ex43	3-2-1	15	4.2	0.1
N3G0Ex44	3-1-2	11	1.1	0.1
N3G0Ex45	1-2-3	7	1.1	0.1

	CHENG'S ALGORITHM			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N4G0Ex01	2-3-4-1	5	0.9	0.2
N4G0Ex02	4-1-2-3	8	1.5	0.2
N4G0Ex03	2-1-4-3	8	0.9	0.2
N4G0Ex04	4-2-1-3	8	1.5	0.1
N4G0Ex05	3-1-4-2	9	1.1	0.2
N4G0Ex06	2-4-3-1	6	0.6	0.2
N4G0Ex07	3-4-1-2	7	1.2	0.1
N4G0Ex08	4-1-2-3	6	0.8	0.2
N4G0Ex09	4-2-1-3	11	1.4	0.2
N4G0Ex10	4-2-1-3	12	1.8	0.2
N4G0Ex11	3-1-2-4	5	0.7	0.2
N4G0Ex12	4-3-2-1	9	1.6	0.1
N4G0Ex13	4-3-1-2	9	1.2	0.2
N4G0Ex14	2-4-1-3	13	3.2	0.2
N4G0Ex15	2-4-1-3	19	2.2	0.2
N4G0Ex16	1-2-3-4	17	3.2	0.2
N4G0Ex17	4-1-2-3	15	1.8	0.2
N4G0Ex18	3-2-4-1	18	1.9	0.2
N4G0Ex19	2-1-4-3	14	1.6	0.2
N4G0Ex20	2-1-3-4	16	1.6	0.2
N4G0Ex21	4-3-1-2	19	2.2	0.1
N4G0Ex22	4-3-1-2	16	1.8	0.1
N4G0Ex23	4-3-2-1	15	2.1	0.1
N4G0Ex24	2-3-4-1	13	1.9	0.1
N4G0Ex25	4-3-2-1	9	0.9	0.1
N4G0Ex26	4-3-1-2	8	1.1	0.1
N4G0Ex27	2-4-3-1	16	3.8	0.1
N4G0Ex28	4-3-1-2	11	2.0	0.1
N4G0Ex29	2-3-1-4	14	1.9	0.1
N4G0Ex30	4-1-3-2	9	1.3	0.1
N4G0Ex31	3-1-2-4	17	2.1	0.2
N4G0Ex32	1-2-3-4	12	1.7	0.2
N4G0Ex33	2-3-4-1	9	1.3	0.2
N4G0Ex34	4-2-3-1	12	1.2	0.2
N4G0Ex35	1-2-4-3	19	3.3	0.2
N4G0Ex36	2-4-3-1	11	1.3	0.2
N4G0Ex37	4-3-2-1	13	2.5	0.2
N4G0Ex38	1-2-3-4	11	1.5	0.2
N4G0Ex39	2-4-1-3	11	1.2	0.2
N4G0Ex40	3-4-1-2	14	1.9	0.2
N4G0Ex41	1-4-3-2	10	2.1	0.2
N4G0Ex42	4-1-3-2	19	2.3	0.2
N4G0Ex43	2-4-3-1	15	1.8	0.2
N4G0Ex44	1-2-4-3	15	2.4	0.2
N4G0Ex45	1-2-4-3	13	1.7	0.2

	CHENG'S ALGORITHM			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N5G0Ex01	3-4-5-1-2	9	2.3	0.3
N5G0Ex02	5-1-4-3-2	9	1.7	0.3
N5G0Ex03	4-3-1-5-2	15	1.7	0.3
N5G0Ex04	5-3-1-4-2	11	2.3	0.2
N5G0Ex05	1-5-3-4-2	13	1.6	0.2
N5G0Ex06	4-5-2-1-3	7	2.2	0.2
N5G0Ex07	5-4-3-1-2	13	1.4	0.2
N5G0Ex08	5-2-4-1-3	17	2.9	0.3
N5G0Ex09	5-1-4-2-3	12	1.7	0.3
N5G0Ex10	2-5-4-1-3	10	2.7	0.3
N5G0Ex11	5-4-3-1-2	18	3	0.3
N5G0Ex12	4-3-2-1-5	13	1.8	0.2
N5G0Ex13	5-4-1-3-2	11	1.5	0.2
N5G0Ex14	1-4-5-3-2	9	1.1	0.2
N5G0Ex15	5-3-4-2-1	20	2.8	0.2
N5G0Ex16	3-5-4-1-2	14	2	0.3
N5G0Ex17	4-2-1-3-5	10	1.3	0.2
N5G0Ex18	5-4-2-3-1	11	1.7	0.2
N5G0Ex19	4-5-2-1-3	20	2.5	0.2
N5G0Ex20	5-3-4-1-2	14	2.2	0.2
N5G0Ex21	5-3-4-2-1	16	1.6	0.3
N5G0Ex22	4-2-3-1-5	13	2.5	0.3
N5G0Ex23	5-1-2-3-4	14	1.7	0.3
N5G0Ex24	5-4-1-2-3	17	1.6	0.3
N5G0Ex25	2-4-3-5-1	9	2	0.3
N5G0Ex26	4-5-1-3-2	12	1.5	0.3
N5G0Ex27	5-4-2-3-1	14	1.5	0.3
N5G0Ex28	3-5-4-1-2	11	2	0.3
N5G0Ex29	2-5-1-3-4	22	3.6	0.3
N5G0Ex30	4-1-5-2-3	10	3.8	0.3
N5G0Ex31	1-5-2-3-4	13	1.4	0.3
N5G0Ex32	5-3-2-4-1	11	1.7	0.3
N5G0Ex33	3-5-1-4-2	26	4.1	0.3
N5G0Ex34	4-3-2-1-5	14	2.8	0.3
N5G0Ex35	4-3-5-1-2	27	4.3	0.3
N5G0Ex36	4-2-3-5-1	14	4.2	0.3
N5G0Ex37	5-3-1-4-2	13	1.9	0.3
N5G0Ex38	5-4-1-2-3	27	3.3	0.3
N5G0Ex39	5-4-3-2-1	27	3.9	0.3
N5G0Ex40	4-5-2-3-1	20	2.7	0.3
N5G0Ex41	3-4-1-5-2	19	3.2	0.3
N5G0Ex42	4-1-5-2-3	21	3.7	0.3
N5G0Ex43	3-4-2-1-5	9	3	0.3
N5G0Ex44	3-2-4-1-5	14	2.5	0.3
N5G0Ex45	3-2-5-4-1	22	2.4	0.3

	CHENG'S ALGORITHM			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N6G0Ex01	4-6-1-3-5-2	12	2.1	0.3
N6G0Ex02	5-1-6-2-3-4	17	2.2	0.3
N6G0Ex03	6-5-3-1-2-4	14	2.2	0.3
N6G0Ex04	1-2-3-4-5-6	15	2.1	0.4
N6G0Ex05	6-4-3-5-2-1	21	2.7	0.3
N6G0Ex06	3-2-6-5-4-1	15	2.3	0.3
N6G0Ex07	5-4-1-3-6-2	15	2.2	0.3
N6G0Ex08	4-2-3-5-6-1	15	2.3	0.4
N6G0Ex09	6-4-3-5-2-1	17	2.8	0.4
N6G0Ex10	6-4-3-5-2-1	15	3.6	0.4
N6G0Ex11	5-6-4-3-1-2	13	2.2	0.4
N6G0Ex12	5-6-4-3-2-1	14	2.2	0.4
N6G0Ex13	5-2-3-4-6-1	10	2.4	0.4
N6G0Ex14	4-6-2-1-3-5	15	3.5	0.4
N6G0Ex15	4-2-6-3-5-1	14	2.6	0.4
N6G0Ex16	4-6-3-5-1-2	13	2.7	0.4
N6G0Ex17	2-6-3-4-5-1	13	2.8	0.4
N6G0Ex18	4-6-2-5-3-1	13	2.2	0.4
N6G0Ex19	6-5-2-3-1-4	13	2.2	0.4
N6G0Ex20	6-5-3-4-2-1	13	2.2	0.4
N6G0Ex21	6-5-4-3-2-1	21	3	0.4
N6G0Ex22	6-2-1-4-5-3	24	3.5	0.4
N6G0Ex23	2-6-3-4-5-1	12	3.1	0.4
N6G0Ex24	6-5-4-3-2-1	11	2	0.4
N6G0Ex25	6-4-1-5-2-3	18	2.6	0.4
N6G0Ex26	6-2-3-5-4-1	13	2.3	0.4
N6G0Ex27	6-5-3-4-2-1	21	3.6	0.4
N6G0Ex28	6-5-2-4-1-3	25	3.4	0.5
N6G0Ex29	4-5-3-1-2-6	22	4.2	0.5
N6G0Ex30	5-3-6-2-1-4	23	4.2	0.5
N6G0Ex31	1-4-6-2-3-5	17	3.5	0.5
N6G0Ex32	6-1-4-5-2-3	19	3.2	0.4
N6G0Ex33	3-6-2-1-4-5	18	2.8	0.4
N6G0Ex34	2-1-5-6-4-3	21	3.5	0.4
N6G0Ex35	6-3-5-2-4-1	15	2.4	0.4
N6G0Ex36	2-3-6-5-4-1	12	2.2	0.4
N6G0Ex37	2-4-1-5-3-6	21	3.9	0.4
N6G0Ex38	5-6-1-2-4-3	20	2.7	0.4
N6G0Ex39	6-5-1-4-2-3	17	2.4	0.4
N6G0Ex40	2-6-5-3-1-4	20	4.2	0.4
N6G0Ex41	5-1-6-4-3-2	21	2.9	0.4
N6G0Ex42	6-1-3-5-2-4	10	1.8	0.4
N6G0Ex43	6-3-2-1-5-4	23	3.3	0.4
N6G0Ex44	2-4-6-1-5-3	17	3.5	0.4
N6G0Ex45	5-6-4-1-2-3	17	2.6	0.4

	CHENG'S ALGORITHM			
	σ^*	k^*	$f(k^*,\sigma^*)$	CPU time
N7G0Ex01	6-3-2-1-7-5-4	15	2.0	0.8
N7G0Ex02	5-3-2-7-1-4-6	15	2.75	0.8
N7G0Ex03	7-1-4-2-3-5-6	17	2.55	0.8
N7G0Ex04	7-2-6-4-1-3-5	18	3.15	0.8
N7G0Ex05	7-6-3-4-5-1-2	21	3	0.8
N7G0Ex06	6-5-4-3-2-1-7	27	3.45	0.8
N7G0Ex07	7-3-6-4-5-2-1	19	3.9	0.8
N7G0Ex08	3-2-4-7-5-1-6	19	3.8	0.8
N7G0Ex09	7-6-5-2-4-1-3	23	3.45	0.8
N7G0Ex10	4-6-7-5-2-1-3	16	2.65	0.8
N7G0Ex11	6-5-4-3-2-1-7	26	3.25	0.8
N7G0Ex12	7-2-1-4-3-5-6	15	2.45	0.8
N7G0Ex13	6-5-4-1-2-7-3	21	3.7	0.8
N7G0Ex14	6-5-2-3-1-4-7	16	3.3	0.8
N7G0Ex15	4-1-5-7-3-6-2	18	3.65	0.8
N7G0Ex16	5-4-2-6-3-7-1	15	2.8	0.8
N7G0Ex17	7-4-6-5-1-2-3	23	3.75	0.8
N7G0Ex18	6-4-5-2-3-7-1	20	3.25	0.8
N7G0Ex19	7-3-4-6-5-1-2	19	2.9	0.8
N7G0Ex20	4-5-7-6-3-2-1	19	3.15	0.8
N7G0Ex21	7-5-6-2-4-1-3	15	2.75	0.8
N7G0Ex22	6-2-4-5-3-1-7	14	2.6	0.8
N7G0Ex23	4-6-5-1-7-3-2	21	3.1	0.8
N7G0Ex24	6-4-5-2-1-7-3	18	3.15	0.8
N7G0Ex25	5-6-7-1-4-2-3	21	3.5	0.8
N7G0Ex26	2-3-5-4 6-1-7	16	2.7	0.8
N7G0Ex27	5-4-2-6-7-3-1	14	2.2	0.8
N7G0Ex28	1-6-4-2-3-5-7	18	2.8	0.8
N7G0Ex29	3-5-2-7-1-4-6	16	2.5	0.8
N7G0Ex30	6-7-1-3-4-2-5	23	3.5	0.8
N7G0Ex31	7-2-1-6-3-5-4	15	2.8	0.8
N7G0Ex32	3-6-7-4-5-1-2	30	10.3	0.8
N7G0Ex33	3-4-2-7-1-6-5	31	5.05	0.8
N7G0Ex34	7-6-2-1-3-4-5	40	5.95	0.8
N7G0Ex35	6-7-4-5-2-3-1	27	6.1	0.8
N7G0Ex36	5-4-3-6-7-2-1	22	3.85	0.8
N7G0Ex37	4-1-2-3-5-7-6	27	3.85	0.8
N7G0Ex38	1-6-4-2-7-5-3	24	4.65	0.8
N7G0Ex39	7-4-3-1-2-6-5	30	5.55	0.8
N7G0Ex40	4-3-6-1-7-2-5	29	3.85	0.8
N7G0Ex41	7-4-3-5-6-2-1	16	2.7	0.8
N7G0Ex42	7-2-1-6-5-4-3	19	3.95	0.8
N7G0Ex43	1-6-2-7-5-4-3	20	3.15	0.8
N7G0Ex44	1-2-4-6-7-3-5	17	2.7	0.8
N7G0Ex45	1-6-7-3-5-4-2	19	3.1	0.8

APPENDIX C

MPS FORMAT

```
NAME        NEOMIP              MINIMISE
ROWS
 N  COST
 G  LIM1
 G  LIM2
 G  LIM3
 G  LIM4
 G  LIM5
 G  LIM6
 G  LIM7
 G  LIM8
 G  LIM9
 G  LIM10
 G  LIM11
 G  LIM12
 G  LIM13
 L  LIM14
 L  LIM15
 L  LIM16
 G  LIM17
 E  LIM18
 E  LIM19
COLUMNS
   E1     LIM1    1 0       COST    1 0
   E2     LIM2    1 0       COST    1 0
   E3     LIM3    1 0       COST    1 0
   E4     LIM4    1 0       COST    1 0
   T1     LIM5    1 0       COST    1 0
   T2     LIM6    1 0       COST    1 0
   T3     LIM7    1 0       COST    1 0
   T4     LIM8    1 0       COST    1 0
   PI1    LIM9    1 0       LIM10   1 0
   PI1    LIM11   1 0       LIM12   1 0
   PI3    LIM13   1 0       LIM14   1 0
   PI3    LIM15   1 0       LIM16   1 0
   PI3    LIM1   -1 0       LIM2   -1 0
   PI3    LIM5    1 0       LIM6    1 0
```

```
    PI4     LIM17    1 0            LIM4     1 0
    PI4     LIM8    -1 0            LIM18    1 0
    PI2     LIM17   -1 0            LIM1    -1 0
    PI2     LIM5     1 0            LIM19    1 0
    P1      LIM9    -1 0            LIM13   -1 0
    P2      LIM10   -1 0            LIM14   -1 0
    P2      LIM19    1 0
    P3      LIM11   -1 0            LIM15   -1 0
    P3      LIM18   -1 0
    P4      LIM12   -1 0            LIM16   -1 0
RHS
    RHS1      LIM1     -0 45        LIM2     -0 45
RHS1          LIM3     -0 45        LIM4     -0 45
    RHS1      LIM5     0 45         LIM6     0 45
    RHS1      LIM7     -0 45        LIM8     0 45
    RHS1      LIM9     0 0
    RHS1      LIM10    0 0          LIM11    0 0
    RHS1      LIM12    0 0          LIM13    0 0
    RHS1      LIM14    0 0          LIM15    0 0
    RHS1      LIM16    0 0          LIM17    0 0
    RHS1      LIM18    0 0          LIM19    0 0
BOUNDS
FX BOUND1   P1     1 0
 FX BOUND1   P4     10 0
 FX BOUND1   P2     3 0
 FX BOUND1   P3     6 0
ENDATA
```

OUTPUT FROM SCICONIC PACKAGE

ITERATIONS	OBJECTIVE	INFEASIBILITIES		TIME
0	0 000000	31 350000(	10)	0 02
7	10 550000	0 000000(	0)	0 03

SOLUTION IS OPTIMAL

NAME	ACTIVITY	DEFINED AS
FUNCTIONAL	10 550000	COST
RESTRAINTS		RHS1
BOUNDS		BOUND1

ROW		AT	ACTIVITY
N	COST	BS	10 550000
G	LIM1	LL	-0 450000
G	LIM2	LL	-0 450000
G	LIM4	BS	6 000000
G	LIM5	BS	4 000000
G	LIM6	BS	1 000000
G	LIM8	LL	0 450000
G	LIM9	BS	9 000000
G	LIM10	BS	7 000000
G	LIM11	BS	4 000000
L	LIM14	BS	-2 000000
L	LIM15	BS	-5 000000
L	LIM16	BS	-9 000000
G	LIM17	BS	3 000000

*** END OF ROWS ***

COLUMN	AT	ACTIVITY
E1	BS	3 550000
E2	BS	0 550000
T4	BS	6 450000
PI1	BS	10 000000
PI3	BS	1 000000
PI4	BS	6 000000
PI2	BS	3 000000
P1	LL	1 000000
P2	LL	3 000000
P3	LL	6 000000
P4	LL	10 000000

*** END OF COLUMNS ***

APPENDIX D

INPUT DATA FOR TEST EXAMPLES FOR BOTH ALGORITHMS

	t_1	t_2	t_3	w_1	w_2	w_3
N3G0Ex01	4	5	3	0.2	0.3	0.5
N3G0Ex02	6	2	1	0.6	0.2	0.2
N3G0Ex03	5	4	9	0.1	0.1	0.8
N3G0Ex04	6	5	7	0.2	0.3	0.5
N3G0Ex05	2	8	4	0.4	0.5	0.1
N3G0Ex06	1	2	5	0.6	0.3	0.1
N3G0Ex07	4	2	3	0.3	0.1	0.6
N3G0Ex08	1	4	5	0.7	0.2	0.1
N3G0Ex09	2	4	3	0.3	0.2	0.5
N3G0Ex10	2	2	6	0.8	0.1	0.1
N3G0Ex11	3	2	5	0.6	0.1	0.3
N3G0Ex12	5	4	6	0.4	0.5	0.1
N3G0Ex13	5	1	3	0.2	0.2	0.6
N3G0Ex14	3	2	7	0.7	0.2	0.1
N3G0Ex15	8	11	9	0.8	0.1	0.1
N3G0Ex16	2	6	4	0.5	0.4	0.1
N3G0Ex17	9	5	8	0.3	0.6	0.1
N3G0Ex18	7	6	9	0.1	0.7	0.2
N3G0Ex19	5	3	7	0.1	0.2	0.7
N3G0Ex20	11	4	7	0.1	0.8	0.1
N3G0Ex21	4	1	5	0.3	0.6	0.1
N3G0Ex22	2	6	7	0.7	0.1	0.2
N3G0Ex23	5	3	1	0.8	0.1	0.1
N3G0Ex24	1	2	4	0.2	0.2	0.6
N3G0Ex25	1	3	2	0.2	0.3	0.5
N3G0Ex26	2	1	5	0.5	0.3	0.2
N3G0Ex27	1	2	2	0.6	0.2	0.2
N3G0Ex28	2	1	2	0.5	0.3	0.2
N3G0Ex29	1	2	3	0.6	0.1	0.3
N3G0Ex30	3	2	1	0.5	0.4	0.1
N3G0Ex31	4	7	5	0.5	0.4	0.1
N3G0Ex32	5	7	4	0.4	0.5	0.1
N3G0Ex33	7	4	5	0.6	0.3	0.1
N3G0Ex34	1	6	3	0.2	0.2	0.6
N3G0Ex35	6	4	8	0.5	0.4	0.1
N3G0Ex36	4	2	4	0.5	0.4	0.1
N3G0Ex37	2	5	3	0.2	0.3	0.5
N3G0Ex38	6	2	4	0.5	0.2	0.3
N3G0Ex39	4	5	1	0.6	0.1	0.3
N3G0Ex40	5	1	3	0.4	0.5	0.1
N3G0Ex41	2	6	2	0.5	0.4	0.1
N3G0Ex42	5	3	1	0.6	0.3	0.1
N3G0Ex43	8	10	5	0.4	0.5	0.1
N3G0Ex44	5	3	6	0.7	0.2	0.1
N3G0Ex45	4	3	2	0.2	0.7	0.1

	t_1	t_2	t_3	t_4
N4G0Ex01	1	2	3	1
N4G0Ex02	3	2	4	5
N4G0Ex03	3	2	5	1
N4G0Ex04	2	5	2	3
N4G0Ex05	5	3	4	1
N4G0Ex06	3	4	1	1
N4G0Ex07	1	2	3	4
N4G0Ex08	2	1	2	4
N4G0Ex09	2	4	2	5
N4G0Ex10	3	4	2	5
N4G0Ex11	2	1	3	1
N4G0Ex12	3	1	5	4
N4G0Ex13	2	1	3	4
N4G0Ex14	1	3	6	10
N4G0Ex15	3	10	4	6
N4G0Ex16	1	7	4	6
N4G0Ex17	1	4	7	11
N4G0Ex18	2	6	12	1
N4G0Ex19	8	5	4	1
N4G0Ex20	5	10	1	2
N4G0Ex21	3	4	6	10
N4G0Ex22	1	4	10	6
N4G0Ex23	4	3	5	10
N4G0Ex24	3	6	7	1
N4G0Ex25	4	1	2	6
N4G0Ex26	1	3	2	6
N4G0Ex27	2	6	4	1
N4G0Ex28	1	3	5	6
N4G0Ex29	2	6	8	5
N4G0Ex30	4	3	1	5
N4G0Ex31	6	2	9	7
N4G0Ex32	5	7	3	4
N4G0Ex33	6	6	3	2
N4G0Ex34	4	5	1	6
N4G0Ex35	8	6	7	5
N4G0Ex36	4	6	2	5
N4G0Ex37	9	3	5	8
N4G0Ex38	7	4	3	2
N4G0Ex39	1	6	3	5
N4G0Ex40	2	7	8	4
N4G0Ex41	6	5	3	4
N4G0Ex42	6	5	4	9
N4G0Ex43	4	7	3	8
N4G0Ex44	9	6	8	5
N4G0Ex45	7	6	5	2

	w_1	w_2	w_3	w_4
N4G0Ex01	0.1	0.3	0.5	0.1
N4G0Ex02	0.3	0.5	0.1	0.1
N4G0Ex03	0.5	0.1	0.1	0.3
N4G0Ex04	0.3	0.5	0.1	0.1
N4G0Ex05	0.6	0.1	0.1	0.2
N4G0Ex06	0.1	0.1	0.7	0.1
N4G0Ex07	0.1	0.1	0.2	0.6
N4G0Ex08	0.6	0.1	0.1	0.2
N4G0Ex09	0.5	0.3	0.1	0.1
N4G0Ex10	0.5	0.3	0.1	0.1
N4G0Ex11	0.5	0.3	0.1	0.1
N4G0Ex12	0.1	0.2	0.5	0.2
N4G0Ex13	0.5	0.1	0.3	0.1
N4G0Ex14	0.1	0.1	0.3	0.5
N4G0Ex15	0.5	0.1	0.1	0.3
N4G0Ex16	0.2	0.5	0.2	0.1
N4G0Ex17	0.5	0.1	0.1	0.3
N4G0Ex18	0.2	0.5	0.2	0.1
N4G0Ex19	0.3	0.1	0.1	0.5
N4G0Ex20	0.2	0.2	0.5	0.1
N4G0Ex21	0.5	0.1	0.3	0.1
N4G0Ex22	0.3	0.1	0.5	0.1
N4G0Ex23	0.1	0.3	0.5	0.1
N4G0Ex24	0.1	0.2	0.6	0.1
N4G0Ex25	0.1	0.6	0.2	0.1
N4G0Ex26	0.1	0.1	0.5	0.3
N4G0Ex27	0.1	0.2	0.3	0.5
N4G0Ex28	0.1	0.1	0.5	0.3
N4G0Ex29	0.2	0.1	0.6	0.1
N4G0Ex30	0.6	0.1	0.1	0.2
N4G0Ex31	0.3	0.5	0.1	0.1
N4G0Ex32	0.1	0.7	0.1	0.1
N4G0Ex33	0.1	0.1	0.7	0.1
N4G0Ex34	0.1	0.2	0.6	0.1
N4G0Ex35	0.1	0.3	0.1	0.5
N4G0Ex36	0.1	0.1	0.1	0.7
N4G0Ex37	0.1	0.1	0.6	0.2
N4G0Ex38	0.1	0.6	0.2	0.1
N4G0Ex39	0.3	0.1	0.1	0.5
N4G0Ex40	0.5	0.1	0.1	0.3
N4G0Ex41	0.1	0.1	0.3	0.5
N4G0Ex42	0.2	0.1	0.6	0.1
N4G0Ex43	0.1	0.1	0.1	0.7
N4G0Ex44	0.1	0.7	0.1	0.1
N4G0Ex45	0.1	0.6	0.1	0.2

	t_1	t_2	t_3	t_4	t_5
N5G0Ex01	1	3	6	10	2
N5G0Ex02	3	4	2	1	6
N5G0Ex03	2	4	5	8	1
N5G0Ex04	1	4	6	2	5
N5G0Ex05	6	3	2	1	5
N5G0Ex06	2	1	4	6	10
N5G0Ex07	1	2	3	4	5
N5G0Ex08	2	4	5	1	10
N5G0Ex09	4	2	6	1	8
N5G0Ex10	2	6	4	5	4
N5G0Ex11	2	4	2	8	10
N5G0Ex12	2	1	6	7	4
N5G0Ex13	1	3	2	5	6
N5G0Ex14	4	2	1	3	2
N5G0Ex15	2	4	2	6	8
N5G0Ex16	2	5	6	1	8
N5G0Ex17	1	3	2	6	4
N5G0Ex18	6	1	2	4	7
N5G0Ex19	1	5	7	8	6
N5G0Ex20	3	4	6	2	8
N5G0Ex21	6	2	4	1	9
N5G0Ex22	2	4	1	6	5
N5G0Ex23	3	2	1	6	9
N5G0Ex24	2	1	4	6	8
N5G0Ex25	3	4	1	5	2
N5G0Ex26	2	4	1	6	3
N5G0Ex27	4	2	1	5	6
N5G0Ex28	2	4	5	1	6
N5G0Ex29	4	10	5	6	8
N5G0Ex30	10	6	7	9	3
N5G0Ex31	6	3	1	2	4
N5G0Ex32	4	1	5	2	6
N5G0Ex33	6	10	11	5	9
N5G0Ex34	3	2	4	10	8
N5G0Ex35	6	8	9	11	7
N5G0Ex36	5	8	7	6	4
N5G0Ex37	3	5	4	2	6
N5G0Ex38	6	4	5	8	9
N5G0Ex39	7	4	6	8	9
N5G0Ex40	6	4	1	8	7
N5G0Ex41	4	8	9	6	5
N5G0Ex42	6	4	8	10	5
N5G0Ex43	4	3	6	8	5
N5G0Ex44	4	5	7	2	6
N5G0Ex45	4	8	9	1	5

	w_1	w_2	w_3	w_4	w_5
N5G0Ex01	0.1	0.1	0.1	0.5	0.2
N5G0Ex02	0.5	0.1	0.1	0.1	0.2
N5G0Ex03	0.5	0.1	0.2	0.1	0.1
N5G0Ex04	0.1	0.1	0.5	0.1	0.2
N5G0Ex05	0.1	0.1	0.5	0.1	0.2
N5G0Ex06	0.1	0.2	0.1	0.1	0.5
N5G0Ex07	0.1	0.6	0.1	0.1	0.1
N5G0Ex08	0.1	0.6	0.1	0.1	0.1
N5G0Ex09	0.6	0.1	0.1	0.1	0.1
N5G0Ex10	0.1	0.1	0.1	0.5	0.2
N5G0Ex11	0.1	0.1	0.1	0.5	0.2
N5G0Ex12	0.1	0.2	0.5	0.1	0.1
N5G0Ex13	0.1	0.1	0.1	0.6	0.1
N5G0Ex14	0.1	0.1	0.1	0.1	0.6
N5G0Ex15	0.1	0.2	0.1	0.5	0.1
N5G0Ex16	0.1	0.1	0.1	0.1	0.6
N5G0Ex17	0.6	0.1	0.1	0.1	0.1
N5G0Ex18	0.1	0.1	0.1	0.6	0.1
N5G0Ex19	0.1	0.6	0.1	0.1	0.1
N5G0Ex20	0.1	0.1	0.6	0.1	0.1
N5G0Ex21	0.1	0.1	0.1	0.6	0.1
N5G0Ex22	0.1	0.6	0.1	0.1	0.1
N5G0Ex23	0.2	0.5	0.1	0.1	0.1
N5G0Ex24	0.6	0.1	0.1	0.1	0.1
N5G0Ex25	0.1	0.2	0.1	0.5	0.1
N5G0Ex26	0.2	0.1	0.5	0.1	0.1
N5G0Ex27	0.1	0.6	0.1	0.1	0.1
N5G0Ex28	0.2	0.1	0.1	0.1	0.5
N5G0Ex29	0.5	0.1	0.1	0.1	0.2
N5G0Ex30	0.6	0.1	0.1	0.1	0.1
N5G0Ex31	0.1	0.6	0.1	0.1	0.1
N5G0Ex32	0.1	0.2	0.5	0.1	0.1
N5G0Ex33	0.6	0.1	0.1	0.1	0.1
N5G0Ex34	0.1	0.1	0.5	0.2	0.1
N5G0Ex35	0.1	0.1	0.1	0.1	0.6
N5G0Ex36	0.2	0.5	0.1	0.1	0.1
N5G0Ex37	0.6	0.1	0.1	0.1	0.1
N5G0Ex38	0.6	0.1	0.1	0.1	0.1
N5G0Ex39	0.1	0.2	0.5	0.1	0.1
N5G0Ex40	0.1	0.5	0.1	0.1	0.2
N5G0Ex41	0.6	0.1	0.1	0.1	0.1
N5G0Ex42	0.2	0.1	0.1	0.1	0.5
N5G0Ex43	0.1	0.1	0.1	0.6	0.1
N5G0Ex44	0.1	0.2	0.1	0.5	0.1
N5G0Ex45	0.1	0.1	0.1	0.1	0.6

	t_1	t_2	t_3	t_4	t_5	t_6
N6G0Ex01	4	3	1	5	2	3
N6G0Ex02	5	1	2	4	8	3
N6G0Ex03	1	2	3	4	5	6
N6G0Ex04	8	4	2	1	3	4
N6G0Ex05	4	2	5	6	1	10
N6G0Ex06	3	5	6	2	1	4
N6G0Ex07	2	4	1	5	8	3
N6G0Ex08	3	5	4	6	1	2
N6G0Ex09	6	4	2	7	1	8
N6G0Ex10	5	3	2	6	1	7
N6G0Ex11	3	4	1	2	6	5
N6G0Ex12	4	2	1	3	6	5
N6G0Ex13	4	3	5	1	7	2
N6G0Ex14	3	1	4	6	5	9
N6G0Ex15	4	3	1	5	2	6
N6G0Ex16	3	4	1	6	2	7
N6G0Ex17	4	5	1	2	3	8
N6G0Ex18	5	3	2	6	1	4
N6G0Ex19	3	2	1	5	4	7
N6G0Ex20	4	3	2	1	5	6
N6G0Ex21	4	3	1	5	7	9
N6G0Ex22	6	7	9	1	2	10
N6G0Ex23	5	8	4	2	3	6
N6G0Ex24	4	3	1	2	3	6
N6G0Ex25	5	2	3	6	1	7
N6G0Ex26	5	3	4	2	1	6
N6G0Ex27	5	4	7	1	6	8
N6G0Ex28	4	3	7	1	10	12
N6G0Ex29	3	4	1	10	12	9
N6G0Ex30	4	1	5	6	8	10
N6G0Ex31	6	2	3	4	5	7
N6G0Ex32	7	3	5	4	2	8
N6G0Ex33	1	4	7	3	5	6
N6G0Ex34	6	8	7	3	5	2
N6G0Ex35	3	1	4	2	5	6
N6G0Ex36	5	6	4	3	1	2
N6G0Ex37	7	8	4	6	2	5
N6G0Ex38	4	1	5	2	8	7
N6G0Ex39	3	2	6	1	5	8
N6G0Ex40	3	7	3	6	8	5
N6G0Ex41	7	3	2	1	8	6
N6G0Ex42	3	1	2	6	1	5
N6G0Ex43	2	4	8	5	3	9
N6G0Ex44	2	6	5	4	3	7
N6G0Ex45	1	2	3	6	7	4

	w_1	w_2	w_3	w_4	w_5	w_6
N6G0Ex01	0.5	0.1	0.1	0.1	0.1	0.1
N6G0Ex02	0.1	0.5	0.1	0.1	0.1	0.1
N6G0Ex03	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex04	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex05	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex06	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex07	0.5	0.1	0.1	0.1	0.1	0.1
N6G0Ex08	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex09	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex10	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex11	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex12	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex13	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex14	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex15	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex16	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex17	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex18	0.1	0.5	0.1	0.1	0.1	0.1
N6G0Ex19	0.1	0.5	0.1	0.1	0.1	0.1
N6G0Ex20	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex21	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex22	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex23	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex24	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex25	0.5	0.1	0.1	0.1	0.1	0.1
N6G0Ex26	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex27	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex28	0.1	0.5	0.1	0.1	0.1	0.1
N6G0Ex29	0.1	0.1	0.1	0.1	0.5	0.1
N6G0Ex30	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex31	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex32	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex33	0.5	0.1	0.1	0.1	0.1	0.1
N6G0Ex34	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex35	0.1	0.1	0.1	0.1	0.5	0.1
N6G0Ex36	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex37	0.5	0.1	0.1	0.1	0.1	0.1
N6G0Ex38	0.1	0.5	0.1	0.1	0.1	0.1
N6G0Ex39	0.1	0.1	0.1	0.5	0.1	0.1
N6G0Ex40	0.1	0.1	0.1	0.1	0.5	0.1
N6G0Ex41	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex42	0.1	0.1	0.5	0.1	0.1	0.1
N6G0Ex43	0.5	0.1	0.1	0.1	0.1	0.1
N6G0Ex44	0.1	0.1	0.1	0.1	0.1	0.5
N6G0Ex45	0.1	0.1	0.1	0.5	0.1	0.1

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
N7G0Ex01	1	3	5	4	2	6	1
N7G0Ex02	2	4	5	3	6	4	1
N7G0Ex03	4	1	2	5	3	6	8
N7G0Ex04	2	6	4	1	3	5	7
N7G0Ex05	2	4	5	1	3	7	9
N7G0Ex06	3	2	4	6	8	9	4
N7G0Ex07	4	2	5	3	4	6	8
N7G0Ex08	4	5	8	6	2	4	3
N7G0Ex09	2	1	3	4	6	8	9
N7G0Ex10	4	2	5	6	3	5	2
N7G0Ex11	4	3	2	5	6	8	7
N7G0Ex12	3	6	2	1	5	4	5
N7G0Ex13	3	2	5	4	6	8	5
N7G0Ex14	2	3	5	4	6	7	4
N7G0Ex15	4	5	3	6	7	2	1
N7G0Ex16	3	5	2	4	6	1	3
N7G0Ex17	2	4	5	6	1	8	9
N7G0Ex18	4	1	3	5	8	7	2
N7G0Ex19	2	4	6	5	3	1	8
N7G0Ex20	4	5	3	6	7	2	4
N7G0Ex21	1	3	4	5	4	6	
N7G0Ex22	2	5	4	3	1	6	4
N7G0Ex23	2	4	3	8	5	6	2
N7G0Ex24	1	2	3	6	5	7	4
N7G0Ex25	1	5	4	3	6	8	7
N7G0Ex26	4	6	5	2	3	1	4
N7G0Ex27	6	2	3	5	7	1	2
N7G0Ex28	6	1	2	7	3	5	4
N7G0Ex29	2	3	8	4	5	6	1
N7G0Ex30	5	4	3	1	6	8	7
N7G0Ex31	3	5	2	6	4	1	8
N7G0Ex32	6	8	10	5	7	9	11
N7G0Ex33	4	9	12	10	8	5	3
N7G0Ex34	8	7	3	9	6	10	15
N7G0Ex35	4	6	8	7	3	11	9
N7G0Ex36	7	3	5	8	9	2	4
N7G0Ex37	8	4	6	9	2	5	4
N7G0Ex38	8	3	5	10	4	6	2
N7G0Ex39	2	6	9	11	08	7	10
N7G0Ex40	3	5	9	11	10	6	2
N7G0Ex41	6	2	4	5	1	3	7
N7G0Ex42	6	5	7	4	2	3	8
N7G0Ex43	9	6	4	3	1	5	2
N7G0Ex44	8	5	2	4	6	1	3
N7G0Ex45	7	3	2	5	1	6	4

	w_1	W_2	w_3	w_4	w_5	w_6	w_7
N7G0Ex01	0.5	0.1	0.1	0.1	0.1	0.05	0.05
N7G0Ex02	0.1	0.5	0.1	0.1	0.1	0.05	0.05
N7G0Ex03	0.1	0.1	0.1	0.5	0.1	0.05	0.05
N7G0Ex04	0.1	0.1	0.1	0.05	0.05	0.5	0.1
N7G0Ex05	0.05	0.05	0.5	0.1	0.1	0.1	0.1
N7G0Ex06	0.1	0.1	0.5	0.1	0.1	0.05	0.05
N7G0Ex07	0.05	0.05	0.1	0.1	0.1	0.5	0.1
N7G0Ex08	0.1	0.1	0.1	0.5	0.05	0.05	0.1
N7G0Ex09	0.05	0.1	0.05	0.1	0.5	0.1	0.1
N7G0Ex10	0.1	0.1	0.05	0.05	0.5	0.1	0.1
N7G0Ex11	0.1	0.1	0.5	0.1	0.1	0.05	0.05
N7G0Ex12	0.1	0.1	0.1	0.5	0.1	0.05	0.05
N7G0Ex13	0.5	0.05	0.05	0.1	0.1	0.1	0.1
N7G0Ex14	0.1	0.1	0.5	0.1	0.1	0.05	0.05
N7G0Ex15	0.1	0.1	0.1	0.1	0.5	0.05	0.05
N7G0Ex16	0.1	0.1	0.5	0.1	0.1	0.05	0.05
N7G0Ex17	0.05	0.05	0.1	0.1	0.1	0.5	0.1
N7G0Ex18	0.1	0.1	0.1	0.5	0.05	0.05	0.1
N7G0Ex19	0.05	0.1	0.05	0.1	0.5	0.1	0.1
N7G0Ex20	0.1	0.1	0.05	0.05	0.5	0.1	0.1
N7G0Ex21	0.1	0.1	0.5	0.1	0.1	0.05	0.05
N7G0Ex22	0.1	0.1	0.1	0.5	0.1	0.05	0.05
N7G0Ex23	0.5	0.05	0.05	0.1	0.1	0.1	0.1
N7G0Ex24	0.1	0.1	0.5	0.1	0.1	0.05	0.05
N7G0Ex25	0.1	0.1	0.1	0.1	0.5	0.05	0.05
N7G0Ex26	0.05	0.5	0.1	0.1	0.1	0.05	0.1
N7G0Ex27	0.1	0.1	0.1	0.1	0.05	0.5	0.05
N7G0Ex28	0.1	0.1	0.1	0.1	0.5	0.05	0.05
N7G0Ex29	0.05	0.05	0.5	0.5	0.1	0.1	0.1
N7G0Ex30	0.05	0.1	0.1	0.05	0.1	0.5	0.1
N7G0Ex31	0.05	0.1	0.05	0.1	0.1	0.5	0.1
N7G0Ex32	0.05	0.1	0.1	0.5	0.1	0.1	0.05
N7G0Ex33	0.5	0.05	0.05	0.1	0.1	0.1	0.1
N7G0Ex34	0.05	0.1	0.05	0.1	0.5	0.1	0.1
N7G0Ex35	0.1	0.1	0.05	0.1	0.05	0.1	0.5
N7G0Ex36	0.1	0.1	0.1	0.5	0.1	0.05	0.05
N7G0Ex37	0.05	0.5	0.05	0.1	0.1	0.1	0.1
N7G0Ex38	0.05	0.1	0.1	0.5	0.1	0.1	0.05
N7G0Ex39	0.1	0.5	0.1	0.05	0.1	0.05	0.1
N7G0Ex40	0.1	0.1	0.5	0.05	0.1	0.05	0.1
N7G0Ex41	0.5	0.1	0.1	0.05	0.1	0.05	0.1
N7G0Ex42	0.05	0.05	0.1	0.1	0.1	0.1	0.5
N7G0Ex43	0.1	0.5	0.05	0.1	0.05	0.1	0.1
N7G0Ex44	0.5	0.1	0.1	0.1	0.05	0.1	0.05
N7G0Ex45	0.05	0.1	0.1	0.5	0.05	0.1	0.1

References

[A96] Mauro Dell'Amico(1996)

"Shop problems with two machines and time lags"

Operations Research Vol 44, No 5

September-October 1996

[ABZ88] Joseph Adams, Egon Balas and Daniel Zawack

"The shifting bottleneck procedure for job shop Scheduling"

[AC91] Applegate D and Cook W (1991) *"A computational study of job shop scheduling"* ORSA J Comput, 3, 149-156

Management Science Vol 34, No 3, March 1988

Printed in U S A

[ALSK96] A Aoki, H Lima, N Sannomiya and Y Kobayashi

"Application of genetic algorithm to a job shop problem with a universal machine"

Proc Of 23rd SICE Intelligent System Symp, pp91-96

[AS93] Ronald G Askin, Charles R Standridge (1993)

"Modelling and analysis of manufacturing systems"

John Wiley&Sons, New York

[AW97] M Azizoglu and S Webster (1997) *"Scheduling job families about an unrestricted common due date on a single machine "* International Journal of Production Research, 1997, Vol 35 , No 5, p 1331-1348

[B69] E Balas *" Machine sequencing via disjunctive graphs An implicit enumeration algorithm "* Oper Res , 17(1969), 941-957

[B74] Baker, Kenneth R (1974) *"Introduction to sequencing and scheduling"* John Wiley&Sons, New York

[B85] Bagchi, U 1985 *"Scheduling to minimize earliness and tardiness penalties with a common due date"* Working paper, Department of Management, University of Texas, Austin

[B87] Bagchi, U 1985 *"Due-date or deadline assignment to multi-job orders to minimize total penalty in the one machine scheduling problem"* Presented at the ORSA/TIMS Joint National Conference St Louis

[BCS87] Bagchi U , Y Chang and R Sullivan 1987 *"Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date"* Naval Res Logist Quart 34, 739-751

[BD90] J E Biegel and J J Davern
" Genetic algorithms and job shop scheduling "
Computers & Industrial Engineering, 19, pp 81-91

[BGG88] Bector, C , Y , Gupta and M , Gupta 1988 "*Determination of an optimal common due date and optimal sequence in a single machine Job Shop*" Int J Prod Res 26, 613-618

[BS90] K R Baker, G D Scudder "*Sequencing with earliness and tardinesss penalties A review*" Operations Research 1990 p 22-36

[BSC86] Bagchi, U , R Sullivan and Y Chang 1986 "*Minimizing mean absolute deviation of completion times about a common due-date*" Naval Res Logist Quart 33, 227-240

[CAT87] Christofides Nicos, Alvarez-Valdes R , Tamarit J M
"*Project scheduling with resource constraints A branch and bound approach*"
European Journal of Operational Research 29 262-273

[C82] J Carlier , " *The one-machine sequencing problem* "
European J Oper Res, 11 (1982), 42-47

[C87] T C E Cheng " *An algorithm for the CON due date determination and sequencing problem* "
Comput Opns Res Vol 14, No 6, pp 537-542, 1987

[C88] T C E Cheng *"Optimal common due date with limited completion time deviation"*
Comput Opns Res Vol 15, No 2, pp 91-96, 1988

[C89] P Chretienne 1989, *"A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints "*
Eur J Opnl Res 43, 225-230

[CC90] J Y Colin, P Chretienne
" C P M scheduling with small communication delays and task duplication "
Technical notes pp 680-684

[DT93] Dell' Amico M and Trubian M (1993) *"Applying taboo search to the job shop scheduling problem"* Ann Oper Research, 41, 231-252

[DLR81] M A H Dempster, J K Lenstra, A H G Rinnooy Kan
" Deterministic and stochastic scheduling " Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems held in Durham, England, July 6-17, 1981
D Reidel Publishing Company

[EC77] Eilon, S and I Chowdhury, 1977 *"Minimizing waiting time variance in the single machine problem"*
Mgmt Sci 23, 567-575

[FMT92] Matteo Fischetti, Silvano Martello and Paolo Toth
" Approximation algorithms for fixed job schedule problems "
Operations Research Vol 40, Supp No 1
January-February 1992

[FS] M A Al-Fawzan and K S Al-Sultan
" A tabu search algorithm for minimizing the makespan in a job shop scheduling "
Institute of Industrial Engineers, 5th Industrial Engineering Research Conference Proceedings

[F82] French S *"Sequencing and scheduling An introduction to the mathematics of the job shop"*
Wiley, Chichester

[GK87] S Gupta and J Kyparisis 1987, *"Single machine scheduling research"* Omega 15, 207-227

[H86] Hall, N 1986 *"Single and multi-processor models for minimizing completion time variance"* Naval Res Logist Quart 33, 49-54

[HP89] Hall, N , and M Posner 1989 *"Weighted deviation of completion times about a common due-date"* Working Paper 89-15, College of Buisness, The Ohio State University, Columbus

[HP91] Hall, N and Posner, M , 1991 *"Earliness-tardiness scheduling problems I weighted deviation of completion times about a common due date"* Operations Research , 39, 836-846

[GLL79] U I Gupta, D T Lee and J Y –T Leung

" An optimal solution for the channel assignment problem "

IEEE Trans Comput C-28, 807-810

[J56] J R Jackson, *" An extension of Johnson's results on the job lot scheduling "* Naval Res Logist Quart 3, 201-203

[K76] A H G Rinnooy Kan 1976

"Machine scheduling problem ",

Classification, complexity and computations

Martinus Nijhoff, The Hague

[K81a] Kanet, J 1981 *"Minimizing the average deviation of job completion times about a common due-date"*

Naval Res Logist Quart 28, 643-651

[K81b] Kanet, J 1981 *"Minimizing variation of flow time in single machine systems"* Mgmt Sci 27, 1453-1459

[KPW94] M Y Kovyalyov, C N Potts and L N Van Wassenhove

"A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work"

Mathematics of Operations Research,

Vol 19, No 1, February 1994

[KGV83] S Kirkpatrick, C D Gelatt, JR and M R Vecchi 1983

"Optimization by simulated annealing" Science 220, p 671-680

[L92] J B Lasserre, "*An integrated model for job-shop planning and scheduling*" Management Science, 38, 8, pp 1201-1211

[LAL92] Peter J M Van Laarhoven, Emile H L Aarts, Jan Karel Lenstra "*Job shop scheduling by simulated annealing*" Operations Research Vol 40, No 1, January-February 1992

[LFS96] H Lima, A Fukui and N Sannomiya "*A method for constructing an autonomous decentralized scheduling system in a parallel machine problem*" Trans of the Institute of Systems, Control and Information Engineers, Vol 9, No 2, pp 97-99

[LPT] C Y Lee, S Piramuthu and Y K Tsai "*Job shop scheduling with a genetic algorithm and machine learning*" International Journal of Production Research 1997 Vol 35, No 4, 1171-1191

[LKSK97] H Lima, R Kudo, N Sannomiya, Y Kobayashi "An autonomous decentralized scheduling algorithm for a job shop process with a multi-function machine in parallel" IEEE 1997

[LM89] E L Lawler, C U Martel *"Preemptive scheduling of two uniform machines to minimize the number of late jobs"* Operations Research Vol 37, No 2, March-April 1989

[MF75] G McMahon and M Florian, *" On scheduling with ready times and due-dates to minimize maximum lateness "* Oper Res, 23 (1975), 475-482

[MSS88] H Matsuo, C J Suh and R S Sullivan *" A controlled search simulated annealing method for the general job shop sceduling problem "* Working paper 03-04-88, Department of Management, The University of Texas at Austin

[NHL82] K Nakajima, S L Hakimi and J K Lenstra *" Complexity results for scheduling tasks in fixed intervals on two types of machines "* SIAM J Comput 11, 512-520

[PS82] C H Papadimitriou and K Steiglitz *" Combinatorial optimization algorithms and complexity "* Prentice-Hall, Englewood Cliffs, N J

[PSS82] Panwalkar, S, M Smith and A Seidmann, 1982 *"Common due date assignment to minimize total penalty for the one machine scheduling problem"* Opns Res 30, 391-399

[PW92] C N Potts and L W Van Wassenhove

" Integratıng schedulıng wıth batchıng and lot-sızıng A revıew of algorıthms and complexıty

J Opnl Res Soc 43, 395-406

[Q87] Quaddus, M 1987 "*A generalızed model of optımal due-date assıgnment by lınear programmıng*" J Opnl Res Soc 38, 353-359

[RS64] B Roy and B Saussman

" Les problemes d'ordonnancement avec constraınts dısjonctıves "

Note DS No 9 bıs, SEMA, Parıs

[S77] Sıdney, J 1977 "*Optımal sıngle machıne schedulıng wıth earlıness and tardıness penaltıes*" Opns Res 25, 62-69

[S79] J G Shanthıkumar (1979) "*Approxımate queueıng models of dynamıc job shops"* A Thesıs submıtted ın conformıty wıth the requırements of the degree of Doctor of Phılosophy ın the Unıversıty of Toronto

[SA84] Sundararaghavan, P and M Ahmed, 1984 "*Mınımızıng the sum of absolute lateness ın sıngle machıne and multı-machıne schedulıng*" Naval Res Logıst Quart 31, 325-333

[SM85] C Santos and M Magazıne

" Batchıng ın sıngle operatıon manufacturıng systems "

O R Letts 4, 99-103

[VR87] Vanı, V and M Raghavacharı 1987 *"Determınıstıc and random sıngle machıne schedulıng wıth varıance mınımızatıon"* Opns Res 35, 111-120

[W78] H P Wıllıams *"Model buıldıng ın mathematıcal programmıng"* Wıley Chıchester, New York, Brısbane, Toronto 1978

[WB95] Scott Webster, Kenneth R Baker (1995) *"Schedulıng groups of jobs on a sıngle machıne"* Operatıons Research Vol 43, No 4, July-August 1995

[WH96] Tohru Watanabe and Yasunorı Hashımoto *" Job shop schedulıng forecastıng margıns to due-dates based-on a neural network "* Advances ın Productıon Management Systems 1996 IFIP

[WTH93a] T Watanabe, H Tokumaru and Y Hashımoto *" Job shop schedulıng usıng neural networks "* Proceedıngs of IFAC World Congress (Ed G C Goodwın And R J Evans), pp 485-488 Pergamon Press, New York

[WTH93b] T Watanabe, H Tokumaru and Y Hashımoto *" Job shop schedulıng usıng neural networks "* Journal of Control Engıneerıng Practıce, Vol 1, No 6, pp 957-961, IFAC

[WTHH92] T Watanabe, H Tokumaru, Y Hashimoto and Y Hirose
" Job shop scheduling based on the estimation of margins to due-dates by using a neural network "
Proceedings of Pacific Conference on Manufacturing,
pp 516-522