ABSTRACT

The study considers the scheduling problem of identical parallel machines subject to minimization of the maximum completion time and the maximum tardiness expressed in a linear convex objective function. The maximum completion time or makespan is the date when the last job to be completed leaves the system. The maximum tardiness is indicated by the job that is completed with the longest delay relative its due date. Minimizing both criteria can help assuring a high utilization of the production system as well as a high level of service towards the client. Due to the complexity of the problem, a Simulated Annealing (SA) heuristic has been implemented to be able to obtain an efficient solution in a reasonable running time.

A set of *n* jobs is assigned, to one of the *m* identical parallel machines. Each job is processed in only one operation before its completion after which it leaves the system. Constraints, such as due dates for each job and setup times for the machines, are considered.

The resolution procedure consists of two phases and begins with an initial solution generator. Then a SA heuristic is applied for further improvement of the solution. 4 generators are used to create an initial solution and 3 to generate neighbour solutions.

To test and verify the performance of the proposed resolution procedure, a computational experimentation has been realized on a set of test problems generated ad-hoc.

CONTENTS

1 ABBREVIATIONS AND SYMBOLS

i:	machine index
m:	maximum number of machines
j:	job index
n:	maximum number of jobs
p_{ij}:	processing time. The period of time that job j needs to be completed in machine i. The notation i will be omitted in two cases; if only one machine can process the job j or if the processing time of job j is independent of the machine i.
r_j:	release date. The earliest moment at which a job j can be processed (this is also known as the ready date).
d_j:	due date. Also known as deadline, which indicates a preferable completion time for job j.
c_j:	completion time. The instant at which the job j is completed.
S_{jk}:	sequence dependent setup time. If the machines have to be reconfigured, or somehow prepared between jobs, and the length of this setup depends on the job just completed and the one about to be started, then the time to realize the setup is sequence dependent. If job j is followed by job k, then the setup time is S_{jk}. Often the setups not only require time; the process also involves a cost because of labour and waste of raw material, e.g.
L_j:	lateness. Indicates whether the job j has exceeded its due date or not when completed.
E_j:	earliness. Indicates the time with which the job j is completed before its due date.
T_j:	tardiness. Indicates the time with which the job j is completed after its due date.
I:	instance. Input data of a scheduling problem.
I_e:	efficiency index. Indicates the performance of a procedure for a proposed solution related to the best solution obtained among all procedures.
D:	set of instances.
D_{Π}:	the whole collection of set of instances.
solinit:	sequence of the initial solution.
solcurr:	sequence of the current solution.
solneigh:	sequence of the neighbour solution.
solbest:	sequence of the best solution found during the search. Schedule presented.

finit:	F_ℓ, or objective function value of the initial solution.
fcurr:	F_ℓ of the current solution.
fneigh:	F_ℓ of the neighbour solution.
fbesthalf:	F_ℓ of the best solution found of SAS when using SASI
fbest:	F_ℓ of the best solution found during the search. In case of using SASI, *fbest* is the F_ℓ of the best solution found of SAI.
it:	counter of iterations.
itmax:	stop criterion in terms of iterations.
same:	counter of ties between *fcurr* and *fneigh*.
*same**:	maximum number of ties allowed.
change:	counter for the change criterion.
*change**:	criterion for the change of method to generate neighbours when using SASI.
terminate:	stop criterion.
SAS:	Simulated Annealing with Swap to generate neighbours.
SAI:	Simulated Annealing with Insert to generate neighbours.
SASI:	Simulated Annealing with Swap and Insert to generate neighbours.

2 INTRODUCTION AND OBJECTIVES OF THE STUDY

2.1 Overview

Scheduling problems, in general, can be understood as the problem of allocating resources over time to perform a set of tasks being parts of a process [23]. Each task compete for resources which can be of different kinds, e.g. manpower, money, energy, machines, tools, etc. The same is seen in task characteristics, e.g. processing time, due date, setup time, which are functions describing the relation between task processing and available resources. Furthermore, a sequence of tasks describing the precedence constraints between them can be defined in different ways as well as criteria, which measures the quality of the performance of a sequence.

The scheduling problem is a decision-making process that is frequently used in a broad range of manufacturing and service industries nowadays [22]. These kinds of decision-making play an important role in different areas of operations, e.g. production, transportation, distribution, information processing and communication. In a company, operating in one of these areas, the planning and scheduling functions rely on mathematical techniques and heuristic methods to allocate its limited resources in such a way that the company optimizes its objectives and achieves its goals.

Take for example the packaging section at a large cosmetics manufacturer. In one part of the facility, different kinds of a shampoo brand are loaded into bottles. The shampoo itself is elaborated elsewhere and arrives to the facility in bulk and the function of the section is to package the product for the final delivery to the client. In the manufacturing facility, three machines are working in parallel to fill the bottles with the product. In total there are 5 different types of shampoo, each with its own bottle. Depending on the demand, the types of shampoo are arriving, packaged and stored at different moments and during longer or shorter time periods. Each time a machine is set to package another type of shampoo some parts of it has to be changed and it has to be loaded with the kind of bottle corresponding to the new shampoo. This change involves a setup time for the machine and may depend on the machine and the product. To be able to serve the demand of the stores, a schedule has to be created that tells the production manager when to package what shampoo and in which machine.

Planning and scheduling are used in many other industries and businesses, in everything from paper mill production and manufacturing of semiconductors, to supply chain and management consulting [22]. Independent of the industry where these tools are used, the overall objective is to minimize the total costs including, e.g. productions costs, transportation costs and inventory holding costs. However, in many environments it may not be immediately clear what impact the planning and scheduling has on a given objective. In practice though, a well-elaborated schedule usually has a measurable impact on system performance and can therefore be used to cut direct and indirect costs significantly. Although minimizing total costs always is a priority to a company, scheduling is not only used to focus on minimizing costs, but also often operated considering other objectives. The objectives may be formulated in such a way that, e.g. in a manufacturing industry, the stock is held at a certain level or so that the clients demands are completely satisfied considering the delivery date.

2.2 General Description of Scheduling

Although scheduling may be used in a number of different industries, when it comes to notation and basic theory, this often refers to the scheduling and planning of a manufacturing system or shop. Nevertheless, the notation and theory is easy adaptable to any given scheduling problem, administrators at a call centre being machines (resources) and customers phoning in being jobs (tasks), e.g. A manufacturing system can be characterized by a number of different factors: the number of resources or machines, their characteristics and configuration, the type of workflow, the material processed etc. To be able to describe these different characters, manufacturing models, ordered into five classes, are used [22]:

- **Project planning**: This model is used whenever a large project with various stages is carried out. Normally there are different activities that may be subject to precedence constraints, i.e. one activity cannot be started until other specific activities are completed. The objective is to minimize the completion time for the latest job or the makespan, and to identify the critical activities that determine the makespan. If anyone of these activities is delayed the whole project will also be delayed.
- **Shop problems**: Depending on the resources, this model consists of scheduling jobs on one or more machines where each job has a route to follow. If there is only one machine or several machines in parallel, the operations are performed on an available machine, while in a job shop a workflow is established, where the

operations are performed on different machines. The operations have to be scheduled to optimize one or several objectives, such as the makespan or the number of early jobs. If all jobs follow the same route, a special case arises called flow shop.

- **Production systems**: In this model an automated material handling is seen where a job normally consists of various operations. As in, for example an assembly line, the movement of the jobs as well as the timing of their processing in the workstations is monitored and controlled. Normally, an objective is to maximize the throughput.
- **Lot scheduling**: In contrast to the previous classes, the production processes in this model are continuous, and are used for medium and long term production planning. The objective is normally to minimize total inventory and changeover costs, the latter originated from the process when a machine is setup to produce another of the existent products.
- **Supply chain**: These models are often a mix of the lot scheduling models and the job shop scheduling models. The objective function often consider inventory holding costs at the different stages of the chain as well as transportation between the stages. Often these models are more complex and have more constraints.

2.3 Research Objectives

The objective is to study the scheduling problem of identical parallel machines to minimize the maximum completion time and the maximum tardiness through a bicriteria objective function.

The study is realized by the implementation of a Simulated Annealing (SA) heuristic. The resolution procedure consists of two phases and begins with an initial solution generator. Then the SA heuristic is applied for further improvement of the solution. 4 generators are used to create an initial solution and 3 to generate neighbour solutions.

To be able to measure the performance of the proposed resolution procedure, the study addresses the following hypotheses:

- **H1**: Any of the proposed procedures for creating an initial solution is better than the others in terms of the value of the objective function of the initial solution.
- **H2**: The proposed procedure, the Simulated Annealing, needs a good initial solution to perform better while dealing with the stated problem considering the bicriteria.
- **H3**: Any of the proposed methods for generating neighbour solutions leads to better results than the others in terms of the value of the objective function of the proposed solution schedule.

2.4 Limitations of Study

The study is limited to the implementation of the Simulated Annealing heuristic to study the stated problem. Furthermore the study is limited to 4 generators to create an initial solution (EDD, SPT, EDD + SPT and Random) and 3 to generate neighbour solutions (Swap, Insert and Swap + Insert).

2.5 Organisation of Dissertation

The study is organized in two parts; the first consists of the introduction of the subject and the basic theory as well as the theory of different procedures used when dealing with the stated problem:

- **Chapter 1**: Explanation to the abbreviations and symbols used.
- **Chapter 2**: Introduces the subject and explains the objectives and the limitations of the study.
- **Chapter 3**: Introduces the shop problem in terms of notation and classification.
- **Chapter 4**: Explains the task of scheduling and gives a focus on parallel machine scheduling and the scheduling with setup times.
- **Chapter 5**: Introduces bicriteria scheduling in terms of definitions, notation and classification.
- **Chapter 6**: Explains different procedures used when solving the scheduling problem from exact algorithms to heuristics and metaheuristics. The chapter also includes some examples of metaheuristics and explains more in detail the heuristic implemented in the study, the Simulated Annealing.

The second part of the report includes the details of the stated problem, the development of the proposed resolution procedure and the computational experimentation. This part ends with the results, the external impact and the conclusions of the study:

- **Chapter 7**: Gives a detailed description of the stated problem, such as its parameters, variables, constraints and the objective function.
- **Chapter 8**: Explains the development of the resolution procedure. The generators for initial solution as well as for generating neighbour solutions are explained. The chapter ends with an overview of the proposed resolution procedure and a summary of the same.
- **Chapter 9**: Accounts for the different phases of the computational experimentation and the results. The chapter begins with an overview to explain the strategy on how to address the hypotheses stated and to measure the performance of the proposed resolution procedure.

- **Chapter 10**: Deals with the external impact of the study, such as the budget and the influence on the environment.
- **Chapter 11**: The last chapter summarizes the study with the conclusions and also indicates possible future work.

3 THE SHOP PROBLEM

3.1 Notation

The shop problem can be defined through two different notations schemes.

1. The first includes four fields, introduced by Conway in 1967, to express the dimension of the problem as well as the machine environment and the objectives [25]:

$$n/m/A/B$$

   - *n*: number of jobs
   - *m*: number of machines
   - A: type of arrangement; setup of machines, workflow, etc.
   - B: efficiency index (or criteria) used.

2. The other notation scheme, which is more commonly used and therefore also the notation used in this dissertation, is the one with three fields, $\alpha/\beta/\gamma$, introduced in 1979 by Graham et al. [20]. Each field of $\alpha/\beta/\gamma$ presents information about the machine environment (α), processing characteristics and constraints (β) and the objectives to be minimized (γ), respectively [22] [19].

   - Machine environment (α):

     - **Single machine** (1): Only one machine is available for the processing of jobs.
     - **Flow shop** (F): Several machines are available in the shop. The jobs in this environment are all following the same processing routing and hence pass the machines in the same order.
     - **Jobshop** (J): Several machines are available in the shop. Each job has the route of its own.
     - **Openshop** (O): Several machines are available in the shop. The jobs do not have fixed routings and can use the machines in any order.
     - **Mixed shop** (X): Several machines are available in the shop and some jobs have their own routing and others do not.

- **Parallel Machines** (P/Q/R): The machines are grouped in one stage and the jobs are mono-operation. For a more detailed description of this particular environment, and the notation P/Q/R, see Chapter 4.2 Parallel Machines Scheduling.
- **Hybrid flowshop** (HF): There are several stages and the jobs all have the same production routing through these stages.
- **General jobshop** (GJ): Each job has a route of its own.
- **General openshop** (GO): The jobs do not have fixed routing.

• Characteristics and constraints (β):

- **Due date** (d_j): Also known as deadline, which indicates a preferable completion time for job *j*. The due dates are a constraint if they are taken into account directly or indirectly, through criteria, in the objective function.
- **Sequence dependent setup time** (S_{jk}): There is a setup time before each operation, which depends on the sequence of operations on each machine, i.e. the machines have to be reconfigured, or somehow prepared before and between processing jobs, and the length of this setup depends on the job just completed and the one about to be started. If job *j* is followed by job *k*, then the setup time is S_{jk}. Often the setups not only require time; the process also involves a cost because of labour and waste of raw material, e.g.
- **Precedence constraints**: In many scheduling problems a job is subject to a precedence constraint, i.e. it cannot start until a set of certain jobs has been completed. These constraints are easiest overviewed through graphs.
- **Machine eligibility constraints**: In some parallel machine environments, any given job can often not be processed in any of the available machines. The job has to be assigned to a machine that belongs to a subset with specific characters and configurations. Often, this situation occurs in an environment with non-identical machines where the machines have different dimensions and/or speeds, e.g.
- **Workforce constraints**: A machine often needs one or more specific operators to process a job due to certain skills required for its implementation. If the workforce is limited in some way, the programmed schedule needs to consider these constraints, which involve an integrated

approach to machine and workforce scheduling.

- **Preemptions**: Restraint in machine processing. If not allowed, a given job must be finished without interruption in a machine once it has been started. If allowed, the process can be interrupted and the semi-elaborated pre-empted job can later be finished in the same or in a different machine at any point in time. This option may include additional setup times, but give rise to the choice of prioritizing and re-prioritizing the sequence even during an ongoing process.

- Objectives to be minimized or optimality criteria (γ):

  - **Makespan** (c_{max}): Also known as maximum completion time and indicates the completion time for the last job to be completed,

$$c_{\max} = \max_j c_j$$

    The makespan is important when having a finite number of jobs and is closely related to the throughput objective. In a parallel machine environment with sequence dependent setup times for example, the minimizing of the makespan also enforces a minimization of the average setup time as well as it balances the workload over the various machines, and therefore it tends to maximize the output rate for the entire system (it indicates a high utilization of machines).

  - **Maximum tardiness**: The maximum tardiness is defined as

$$T_{\max} = \max_j T_j$$

    and is closely related to the maximum lateness, indicating the performance of the schedule. In the industry, minimizing this objective is used to control the quality of the service to the customer concerning the deliverance of each product and its pre-established delivery date.

  - **Throughput**: In many industries maximizing the throughput is the most important objective and the companies have different ways of measuring how well they do so. The throughput of a system, or its output rate, is often determined by machines that have lower capacity then the rest, i.e. bottlenecks. Maximizing the output rate of these machines will also maximize the throughput of the entire system. The most important issue is to make sure that these machines do not have any idle time, i.e. that there are always jobs waiting in the queue for these machines. In the case of having sequence dependent setup times, the average setup time should also be minimized at those bottlenecks.

- **Maximum lateness**: The maximum lateness is defined as

$$L_{\max} = \max_j L_j$$

and minimizing this objective is equal to minimizing the schedule's worst performance.

- **Flow time**: The sum of the total completion time for all the jobs scheduled given by

$$\sum_{j=1}^{n} C_j$$

This objective minimizes the average number of jobs in the system and is closely related to the objective of minimizing the average throughput time and the work-in-progress (WIP) inventory.

- **Number of tardy jobs**: This is a responsive statistic in databases and it is often used to measure the percentage of on-time shipments. However, the objective focus only on whether the job is tardy or not, not on how tardy it is. In practice, this can lead to schedules that contain few tardy jobs, but those that are tardy can be so to a large extent, which may be unacceptable.

- **Total tardiness**: The sum of the total tardiness for all the jobs scheduled given by

$$\sum_{j=1}^{n} T_j$$

which deals with the problem mentioned when using the number of tardy jobs. This objective may result in a schedule that has the same total tardiness as one that focuses on the number of tardy jobs, but the tardiness of each job is more evenly distributed.

Since a bicriteria is implemented through the linear convex objective function, the notation presented in this chapter is not sufficient to completely term the problem stated. Further notation when dealing with multicriteria scheduling is presented in Chapter 5 Bicriteria Scheduling.

3.2 Problem Classifications

The shop problem is classified in three different categories [25]: static, semi-dynamic and dynamic.

3.2.1 Static Problem

- The number of jobs are known and limited and are processed in a shop with a limited number of machines.
- When the sequencing is realized the route of each job is known and its corresponding operations, as well as in which machine the operations are performed and the time of each operation.
- All the jobs and machines are available at the same time, often termed 0.
- The goal is to find a schedule that is able to optimize an established objective.

3.2.2 Semi-dynamic Problem

- The number of jobs are known and limited and are processed in a shop with a limited number of machines.
- When the sequencing is realized the route of each job is known and its corresponding operations, as well as in which machine the operations are performed and the time of each operation.
- The jobs and machines are most likely available at different moments, these moments are however always known.
- The goal is to find a schedule that is able to optimize an established objective.

3.2.3 Dynamic Problem

- The operation of the shop is not limited in time.
- The number of jobs is also unlimited. The characteristics of the jobs are not defined in a given moment but rather while the operation of the job-shop proceeds.
- The characteristics of a job are defined in the moment when the job arrives at the job-shop.

- As the operation of the shop proceeds, some jobs are completed and leave while others arrive to be processed.
- Since the situation in the shop changes with time, the determined schedule should be re-elaborated through determined cycles.
- The goal is to establish a scheduling procedure and verify its quality; the objectives are calculated through mean values produced during a sufficiently long period of time.

4 SCHEDULING

4.1 Introduction

Scheduling is a procedure that tries to define [25]:

- In which available resource or machine should the necessary operations be realized so that the demanded jobs can be delivered.
- The moments in time (dates) when this action should take place.

The task of scheduling could be broken down into three sub-tasks:

- The **load**, that consists of assigning the resource to the corresponding operation where the operation should be realized.
- The **sequencing**, that defines the order or the sequence of the operations assigned to the same resource (an example given in Figure 4.1). When dealing with a shop problem the sequencing should consider:
    - *n* jobs, *m* machines.
    - The completion of each job, i.e. the process of a series of determined operations in a determined order.
    - The determined and known length of each operation assigned to the *m* machines.
    - The determination of a schedule of operations in every machine and the interval of the realization of the operations with the objective to optimize an index that measures the efficiency of the schedule.
- The **timing**, establishes the dates of the realization.

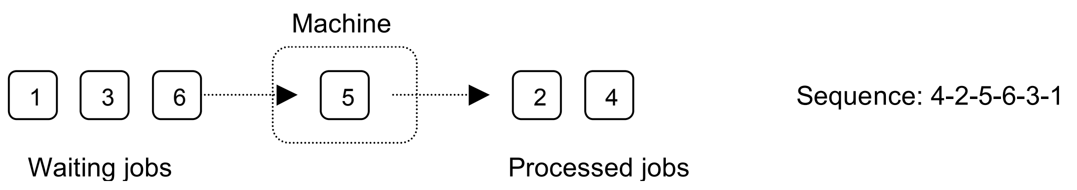

Figure 4.1. Sequencing of jobs in a machine.

4.2 Parallel Machine Scheduling

4.2.1 Definitions

There exist three different machine environments when dealing with parallel machines [22]:

- **Identical machines in parallel** (P_m): A number of m ($m \geq 2$) machines available in parallel where the jobs can be processed in any of the machines. Each job j will have the same processing time p_j in any of the machines. In many production environments, there are several stages or work centres, consisting of machines in parallel. If a work centre is a bottleneck, the parallel machine models can be used in the same way as the single machine models, to analyze the performance of the entire system by modelling the bottleneck separately.
- **Machines in parallel with different speeds** (Q_m): This setup is also known as uniform machines, where there are m ($m \geq 2$) machines available in parallel with different speed, v_i. The time p_{ij} that the machine i takes to process job j is equal to $\frac{p_j}{v_i}$.
- **Non-identical machines in parallel** (R_m): A more general situation than the previous case with m machines in parallel, staged so that machine i can process job j at speed v_{ij}. The time p_{ij} that the machine i takes to process job j is equal to $\frac{p_j}{v_{ij}}$.

The parallel machine scheduling problem consists of two sub problems:

- Assigning jobs to machines.
- Sequencing the assigned jobs on the machine.

The assigning task of scheduling when dealing with identical parallel machines is illustrated in Figure 4.2. There are various procedures to use when creating a schedule on parallel identical machines. To what extent the determined objectives are achieved, depends on the quality of the schedule and therefore also of the strength and the appropriateness of the applied scheduling procedure [11]. More on procedures to use when scheduling is presented in Chapter 6 Resolution Procedures.

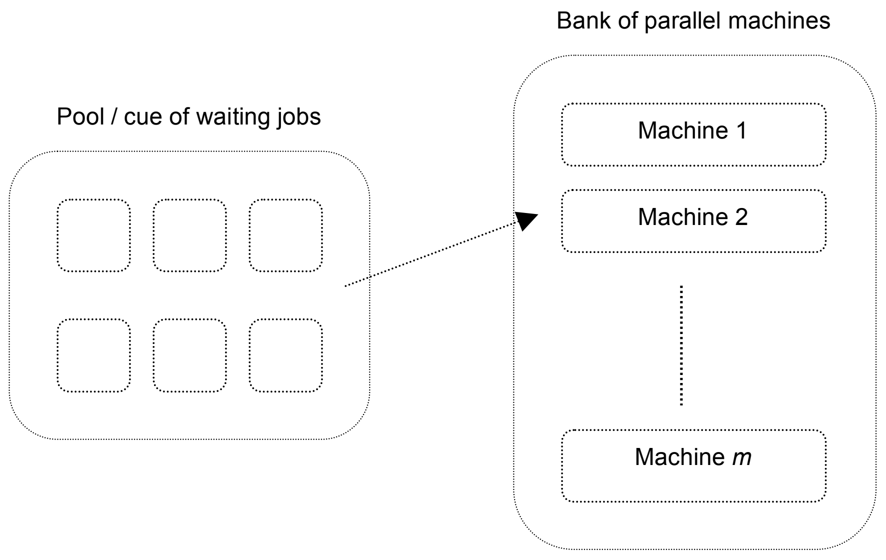

Figure 4.2. Overview of the task presented when creating a schedule with parallel machines.

4.2.2 Relevant Papers on Parallel Machine Scheduling

A lot of research has been made on parallel machines, independent of the machine environment, i.e. on identical parallel machines, unrelated, etc. The papers presented here have some or a lot in common with the stated scheduling problem of this dissertation, being the machine environment, criteria for the objective function or perhaps the technique used to solve the problem.

Kim et al. (2002) [27] studied unrelated parallel machines with setup times using the same heuristic as in this study, the Simulated Annealing algorithm. The paper deals with a real problem encountered in a semiconductor production facility where a part of the manufacturing process suffers from a bottleneck. The problem is solved using different techniques and the result shows that the adapted SA algorithm design for the problem presented outperforms traditional neighbourhood search methods.

Another paper on parallel machines with setup times is presented by Lee et al. (1997) [24]. The problem is solved in phases also using the simulated annealing algorithm to explore the solution space. As criterion, the total weighted tardiness is used, and the result shows an algorithm easy to apply to more complex problems, which has also been implemented in a number of real situations.

A more recent paper that illustrates a scheduling problem not often seen in the literature is presented by Ravetti et al. (2007) [7]. The problem considers sequence and machine dependent setup times with unrelated parallel machines. The problem is resolved by a mathematical model based on the metaheuristic GRASP and the study shows a simple and flexible approach to solve this kind of situations.

A well respected and often mentioned paper is the state of the art review given by Cheng et al. (1990) [9]. The paper relates to the whole class of scheduling problem that consists of parallel machines in different environments.

4.3 Scheduling with Setup Times

In real industry, if a machine has to manufacture a different type of product, a reconfiguration or preparation of that machine or system is almost always required [22]. The setup times considered may or may not be sequence dependent, i.e. the time to prepare the system may or may not depend on the last product that was manufactured and the next one about to be processed. Take an airport for example; a plane has to be prepared between trips. After landing, the passengers have to get off, the plane has to be cleaned and refuelled and the new passengers have to board the plane before the takeoff. The time between two takeoffs may be fixed or random depending on the type of the plane that is used. Furthermore, at an airport, the time between to takeoffs may vary. Each plane taking off causes turbulence and it is necessary to keep the runway idle for a couple of minutes. This idle time will be longer if a smaller plane takes off after a larger one than vice versa, because a larger one causes more turbulence and a smaller one is more affected by turbulence.

Most often, scheduling problems consider sequence dependent setup times rather than fixed setup times, since such situations do not occur just as frequently in real industry related problems [11]. There are several classes of models of setup times (sequence dependent, fixed, Charles-Owaba and Lambert, etc), however, in the stated problem only one will be considered, the one with sequence dependent setup times.

An interesting paper, which considers setup times, is presented by Cheng et al. (2004) [3]. The paper is presented more as a survey explaining the different classes of problems considering setup times, and also presents a framework for the different models and different heuristics used. Another more extensive and recent survey on the subject is presented by Allahverdi et al. (2008) [10], where all classes of problems are mentioned and the techniques used to solve them.

5 BICRITERIA SCHEDULING

5.1 Introduction

Since the first paper on scheduling was published in 1954, many variants of the scheduling problem have been formulated by changing machine environment, side-constraints and objective functions [18]. During many years, it was common practice to take only one performance criterion into account in the objective function. However, in real world industry, when the quality is measured, multiple criteria are what really reflect the performance. In a manufacturing plant, for instance, the planning phase of the production consists of three different levels: strategic, tactical and operational. On the tactical level, the quantities of products to make by time period is determined and the emphasized objectives are [19]:

- To satisfy the requirements of the client, i.e. to deliver the product wanted, in the desired quantity and at the desired date.
- To balance continuously the resources necessary for production by avoiding underloading as well as overloading.
- To ensure the maximum profitability of the production.

At the operational level, the established plan must be followed as good as possible, however, this gives rise to some coherence problems since the plan on the tactical level is made up of aggregated information but on the operational level the information is detailed. Scheduling has as principal objectives:

- To minimize work-in-progress in the shop.
- To give high priority to the promised delivery dates given to the clients.
- To optimize the shop resources.

Given these objectives, the scheduling problem is in its nature, concerning the production, very often multicriteria. If only one criterion is emphasized, the general performance is likely bad since the production is not balanced considering each objective. If the highest priority for example is given to the deliverance of products to the clients, the inventories are likely to be large and expensive and vice versa. In order to reach a compromise, the development of the area of multicriteria scheduling arose, to give the possibility to measure the quality of a solution on all the important criteria.

If all the different types of scheduling problems are considered, however, relatively little research has been conducted to evaluate and compare the performance of heuristic algorithms in finding the set of efficient solutions for multiple criteria scheduling problems [13]. The reason why will almost certainly be found in the complex nature of these problems, even though they are a more appropriate description of most shop floors. However, there are two surveys reflecting the subject and they are worthwhile mentioning; Hoogeveen (2005) [18] and T'Kindt and Billaut (2002) [19].

5.2 Definitions

Multicriteria scheduling consists of computing a so-called *Pareto optimal* schedule for several conflicting criteria. A Pareto optimal schedule is an all efficient schedule in which any improvement of the performance with respect to one of the criteria causes deterioration with respect to one of the other criteria. The task can be broken down into three phases [19]:

1. **Modelling of the problem**. This permits determining the nature of the problem as well as the definition of the criteria to be taken into account.
2. **Taking into account the criteria**. The way in which the criteria are taken into account, also called a module, is defined.
3. **Scheduling**. An algorithm for solving the problem is presented, also called resolution module, which finally leads to the solution of the problem.

The first phase consists of three parts:

- Defining the relevant conflicting criteria, which have to be taken into account, i.e. it is assumed that minimizing one criterion is not equivalent to minimizing another.
- Defining the environment where the scheduling problem occurs, e.g. the number of machines, the organisation of the shop etc.
- Defining the particular constraints of the problem, e.g. authorizing preemptions or not, release dates, etc.

These definitions make it possible to draw up a model of the scheduling problem in hand.

The second phase is normally the most difficult one. Depending on how the decision maker perceives the problem, the objective to achieve for each criterion, is established, expressed as the objective function as well as the method to approach the problem and the resolution procedure.

The last phase has the objective to provide a schedule, which optimizes the objective function defined in the earlier phase.

5.3 Notation

A multicriteria scheduling problem can be noted in a general way by using the three-field notation introduced by Graham in 1979, where the γ field contains a list of the criteria:

$$\alpha / \beta / Z_1, Z_2, ..., Z_K$$

The γ field, representing the different criteria considered, is most commonly defined by [19]:

- $F_\ell(Z_1, ..., Z_K)$. The objective is to minimize a linear convex combination of the *K* criteria, where each criterion has an allocated weight indicating its importance.
- $Lex(Z_1, ..., Z_K)$. Trade-offs are not allowed between criteria. Under a lexicographical approach, the criteria are ranked in order of importance where the first criterion is optimized first, the second criterion afterwards, subject to achieving the optimum with respect to the first criterion, and so on.
- $GP(Z_1, ..., Z_K)$. If there are goals to reach for each criterion the problem is not to optimize the criteria, but to find a solution that satisfies the goals.

5.4 Classification of Resolution Procedures

When solving a multicriteria scheduling problem the characteristics of the problem defined in the two first phases will lead to the development of a resolution algorithm. Depending on the characteristics, three different cases of resolution algorithms can be distinguished [19]:

1. **Priori method**. The tool used, the algorithm, returns a unique solution. An instance of a scheduling problem is the input, and together with the selected values of the parameters, a resolution of the scheduling problem can be obtained and returned.

2. **Interactive method**. The decision maker wishes to intervene and the algorithm will therefore ask for new search directions. Once provided, a calculated solution is proposed and if the decision maker wishes to proceed in another search direction, the process is repeated.
3. **Posteriori method**. The resolution procedure varies the parameters so that a set of solution schedules is presented. The decision maker then selects one schedule from the set that best fits the requirements.

6 RESOLUTION PROCEDURES

The scheduling problem originates from real-world, every day situations and industrial processes and the desire to control and optimize their performance has given birth to the formulation and solution of mathematical models describing these situations [11]. Since almost any real-world situation is peculiarly complex in its nature, the mathematical model describing it should be constructed sufficiently complex to include all key aspects of the problem, yet simple enough to be able to provide a good and understandable solution.

In any mathematical model describing a real-world situation generally and a scheduling problem particularly, different aims are identified and typically expressed as a mathematical objective function. The value of this objective function represents the quality of the solution and is also the target for improvement. Since most mathematical models cannot fully describe the real, often highly complex and constrained problem, the most important objective is to find a feasible solution. In the attempt to find feasible solutions and to improve the value of the objective function, subject to the relationships and constraints arising from the model chosen, a solution methodology or procedure is developed. The procedure often includes basic elements such as the creation of an initial solution, how to proceed and finding other solutions and how to evaluate them in terms of the objective function.

There are many solution procedures, each one with its advantages depending on the type of problem and established mathematical model. In the following sections, two categories of solution procedures are mentioned in relation to scheduling problems.

6.1 Exact Algorithms

When a mathematical model can be created to exactly render the problem to be solved and consider all of its features, the best way is to use an exact algorithm as a solution procedure [15]. Such models include for example linear programming, integer programming and nonlinear programming models. Through its search, the model will consider each and every possible solution until all of the feasible solutions have been examined. Then it will choose the best one, guaranteeing it to be the optimal solution to the problem. The model could also search for the best solution through a specific search path, excluding solutions that for sure are not the optimal one. Whatever the model, an exact algorithm can always find and guarantee the optimal solution to the problem (or guarantee that there is no optimal solution to be found). Examples of such models are the simplex method and branch-and-bound.

However, the difficulty in scheduling problems lies in their combinatoric nature [11]. When determining a sequence of jobs that are to be processed in a machine, one of the jobs already sequenced can be selected and put into another position in the sequence, creating a new solution schedule (see Figure 6.1). Finding the correct order in which to process the jobs involves making a selection from an extremely large population of solutions. For example, a scheduling problem that involves sequencing 20 jobs can have as many as 20! potential solutions. This means approximately $2.43 \cdot 10^{18}$ different possible combinations, and yet it is a scheduling problem so small and most uncommon in the real industry. Even though it is possible to enumerate all possible solutions when the size of the scheduling problem is very small, the time required for this approach increases extremely fast and soon becomes unreasonable when the scheduling problem adopts a meaningful size. Effective algorithms used for scheduling problems are therefore constructed in a way that allows them to find good solutions within a reasonable amount of time. The approach with an exact algorithm, however, has also been investigated within the field of scheduling problems, and some of the relevant papers are mentioned below.

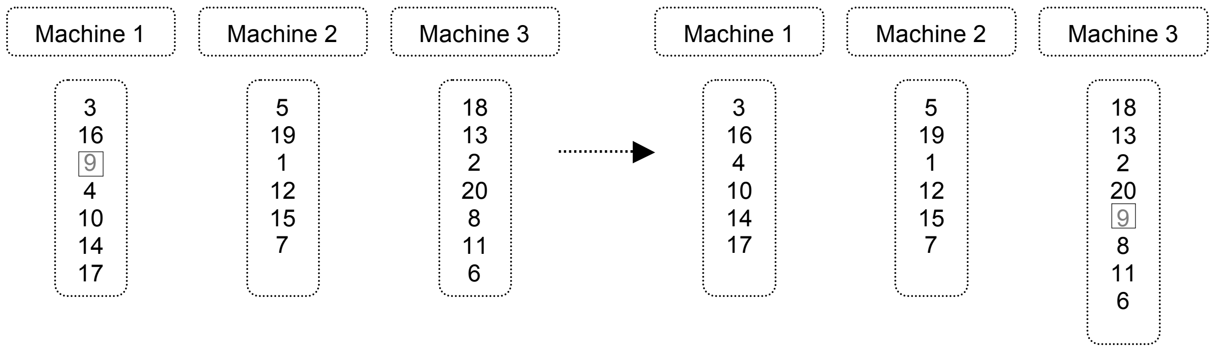

Figure 6.1. A schedule transformed into another by a simple job movement.

An approach using an exact algorithm is the branch-and-bound procedure, an example given by Lomnicki (1965) [2] and P. Brucker et al. (1999) [1]. A more recent paper on the scheduling problem, featuring identical parallel machines and the approach using an exact algorithm, is presented by E. Mokotoff (2004) [6]. A linear model is built, including pre-processing phases, and tested with successful outcome on various situations.

6.2 Heuristics

A heuristic method is a procedure that is likely to find a good feasible solution but leaves no guarantee of its quality or whether it is optimal or not [15]. All the possible solutions are not considered, since that would require an infinite amount of time, but rather a part of the solution space with solutions that might or might not be optimal. The solution space is searched smarter, discarding those parts that certainly not will contain good solutions and focusing more on those parts that could include a good one. Nevertheless, a well-designed heuristic method can often provide a near-optimal solution, or indicate that no optimal solution exists. The method should also be efficient enough, so that it can deal with large problems within a reasonable time. The procedure is generally a fully developed iterative algorithm, where in each iteration it strives to find a solution that is better than the best one found previously. After terminated, when a stop criteria is satisfied, the solution provided is the best one found during any iteration. The search methods used in each iteration are often based on simple common sense and the heuristic procedures are often *ad hoc*, i.e. each method is designed to adjust to a specific problem rather than being a general solution method.

6.3 Metaheuristics

6.3.1 Introduction

The problem with ordinary heuristic methods is that for every problem given, a procedure must be designed to fit and to solve the problem [15] [14]. However, in recent years another type of procedure has been developed, the metaheuristic, that consists of both a general structure and strategy guidelines to adjust to the specific problem given. This approach is very timesaving and metaheuristics have become an important tool for solving a wide range of practical problems.

Furthermore, ordinary heuristics often are local improvement procedures, i.e. they try to improve the current solution within the local neighbourhood of that solution. This means that for every iteration, the method will find a solution near the current one and accept it if its better, converging towards the local optimum within the neighbourhood of the starting solution. The drawback of this approach is that if the given problem consists of multiple local optima, the procedure applied will converge to one local optimum and then stop. Depending on where the procedure begins the search, it could find different local optima, finding the global optimum only if the search happens to begin within the local neighbourhood of this global optimum. A way of overcoming this drawback is to start the heuristic procedure a number of times from different, randomly chosen initial solutions. Each repetition will often lead to a new local optimum, and with a sufficient number of repetitions, the best local optimum found might also coincide with the global optimum. However, this approach does not work well on large problems, with a complicated feasible solution region and a large number of local optimum that will make it complicated and difficult to reach the global optimum by randomly selecting initial solutions and search within each local neighbourhood.

For larger and more complicated problems, metaheuristic procedures are used, where the search method combines local improvement procedures with more advanced and intelligent strategies to create a process capable of escaping from local optima and performing a robust search of the solution space. An important characteristic of the metaheuristic is hence the possibility to escape from a local optimum after reaching it, and depending on the method, there are different ways of doing so. One common way, besides searching locally for better solutions than the current one, is to also have the possibility to accept a neighbour solution if it is worse than the current one. Another way is to prohibit solutions, within the current solution neighbourhood, to force the procedure to search in another direction away from a local optimum.

No matter the technique used to escape from a local optimum, all metaheuristics tend to follow a typical search pattern. When applied to a problem, the procedure starts to search the neighbourhood for the local optimum. But rather than being trapped there, the procedure then guides the search towards worse solutions to be able to find other neighbourhoods to explore. Even if a local optimum, found early in the search, happens to be the global optimum, the heuristic algorithm continues its search trying to verify this fact before stopping. In theory, the search process could continue forever, unless the optimal value of the problem at hand is known. However, for practical reasons, a stopping criterion is established that determines when the algorithm has found a good-enough solution. The most common stopping criteria are:

- After a fixed number of iterations (or a fixed amount of CPU time)
- After a number of iterations without an improvement of the objective function value
- When the objective reaches a pre-specified threshold value.

The great advantage of metaheuristic algorithms is that they often move quickly towards very good solutions even if the problem given is a large and complicated one. Hence it is a very efficient approach, however the best solution provided leaves no guarantee whatsoever of being optimal or even near optimal. Therefore, if the problem given requires to be solved optimally, an algorithm that can provide that solution should be executed. However, if the problem is too complicated to be solved by an exact algorithm, as in most cases of combinatorial optimization (for example scheduling problem, the travelling salesman problem, vehicle routing problem, etc), metaheuristics are an appropriate approach.

6.3.2 Initial Solution Heuristics

One characteristic feature of many heuristic and metaheuristic procedures is the process of generating neighbour solutions starting from an initial solution. The most common way of creating this initial solution is by a total random order. There are, however, also rules to apply to perhaps position the initial solution well in the solution space giving the procedure a good start so that it could easier come to a better result in the end. When dealing with scheduling problems, two of the most common rules are SPT and EDD [22]:

- **SPT:**

  SPT or Shortest Processing Time. When applying this rule, the jobs are ordered in a list by their processing time p_j, where the job with the shortest time needed is assigned first and the job with the longest time last. This rule has been shown to minimize the average number of jobs waiting to be processed and also the average stock level in a manufacturing facility for example.

- **EDD:**

  EDD or Earliest Due Date. The jobs are here ordered in a similar way as in SPT, but now according to the moment when they are expected to be completed, d_j. The rule prioritizes hence the most urgent jobs and has been shown to minimize the maximum lateness of all jobs, $L_{\max}$.

Once an initial solution has been created, the metaheuristic starts generating neighbours to explore the solution space in its search for a better solution. The most common improving metaheuristics used when dealing with combinatorial optimization problem are presented in continuation.

6.3.3 Tabu Search

Tabu Search (TS) was first presented by Fred Glover in 1986 [14]. This metaheuristic applies local search procedures to converge to local optimum solutions, but also uses a short-term memory, called a tabu list, where it stores a number of previously visited solutions. The prevention of cycling back to these solutions, temporally present in the tabu list, allows non-improving moves and guides TS away from local optima and towards unexplored neighbourhoods. The concept of recording the recent history of the search, and adapting it continuously, links the method to the philosophy of Artificial Intelligence.

The use of this evolving memory in TS is the key to its sophisticated strategy; when finding a local optimum, something has to be done to prevent the search from tracing back its steps to where it came from. This is achieved by disallowing moves that reverse the search route, by declaring tabu recently evaluated solutions. The number of iterations, during which a solution stays in the tabu list and hence a search move back to that particular solution is prohibited, is called tabu tenure of the move. Usually, only a fairly limited quantity of information is recorded in the tabu list, however there are several possibilities regarding its content. Normally, the information stored includes the last few transformations performed on the current solution and thereby prohibits the reverse transformations. Other tabus are based on key characteristics of the solutions themselves or of the moves, and the tabu information should always be well adapted to the way the search is performed (the move to a neighbour solution). Furthermore, the tabu lists are often not fixed in length as this can result in solution cycling, and the simultaneous use of different tabu lists is common.

Another feature that well-designed TS algorithms have is that of the aspiration criterion. The aspiration criteria are algorithmic techniques that allow one to cancel tabus from the list. This is necessary because, even though it is essential to TS, the tabus can become too efficient, and this may lead to prohibition of attractive moves, even if there is no risk of cycling. Tabus can also block ways out from local optima, leading to a stagnation of the search process. Even though there exist very complicated and well-designed aspiration criteria, the most common one consists in allowing a move, even though it is tabu, if it results in a solution with an objective function value better than that of the current best-known solution. This solution has obviously not been visited previously, and the solution itself is not tabu, but the move to get there could be.

To make a TS algorithm fully effective, more elements have to be implemented into the algorithm. One element commonly used is that of intensification, with which parts of the solution space that seem more promising, are more extensively explored. Another element is that of diversification that forces the search into previously unexplored areas of the search space.

6.3.4 Genetic Algorithms

The term Genetic Algorithm, or GA, was first used by John Holland in 1975 [14]. The common denominator for genetic algorithms, because there are a large number of them with different variations, is the use of the concepts of mutation and selection, the core of the neo-Darwinian theory of evolution. Because of the somehow abstract search procedure, a different kind of representation is acquired when using GA. Each candidate solution, or phenotype, has its abstract representation, called genotype, which normally consists of a binary string. The search process, or evolution, starts with a number of randomly generated phenotypes, the population, and proceeds through generations, in which the fitness of the phenotypes is evaluated. In each iteration, various phenotypes are selected based on their fitness and they are recombined to form a new population.

In GA, the generation of new solution candidates are the one of the key features of the method. The two most common ways of generating new populations are crossover and mutation. When using crossover, an offspring is created combining genes from two parents. This is done by replacing some of the genes in one parent with the corresponding genes of the other. In practice, since each phenotype is a binary string (or similar), a part of the string of one phenotype is combined with the corresponding part of another phenotype. In Figure 6.2, an example is given to demonstrate a one-point crossover (crossover position 3, resulting in two offsprings O1 and O2 from the parents P1 and P2).

```
P1: 001100          O1: 001011
P2: 110011          O2: 110100
```

Figure 6.2. One-point crossover.

The other way of generating new phenotypes is mutation, in which a number of genes is randomly chosen and their value is altered somehow. In the case with binary strings, the value of the gene is simply changed from being 0 to 1 or vice versa. In the next example, seen in Figure 6.3, it is shown how a mutation is used to generate a new phenotype (the parent P1 undergoes a mutation in the positions 2 and 5 resulting in the offspring O1).

```
P1: 001100          O1: 011110
```

Figure 6.3. Mutation of parent to generate a new phenotype.

As GA is a stochastic search method, and not a simple neighbourhood search algorithm that stops when reaching a local optimum, in theory it could run forever if not given a certain termination criterion. Commonly, a GA is set to stop after having generated a number of populations, after having reached a determined CPU time or when reaching a satisfactory fitness level among the phenotypes. As in the case of all metaheuristics, one cannot guarantee that the best phenotype, or solution, obtained is optimal. However, it is of most importance, to be able to achieve a good feasible solution, that much work is put into the design of the initial population and its representation.

6.3.5 Simulated Annealing

Introduction & Background:

Simulated Annealing (SA) was developed in 1983 to deal with difficult nonlinear problems and belongs to a class of local search algorithms that are known as threshold algorithms [16]. These algorithms are widely used because they can often easily achieve very good solutions to a broad range of practical problems. Furthermore, SA is especially popular because of its stochastic component, which facilitates certain mathematical analyses. The algorithm is often easy to implement, has distinct convergence properties and uses hill-climbing moves to escape local optima [14]. It is not used much for continuous optimization problems, but rather for discrete ones.

The name, Simulated Annealing, originates from its analogy to the thermodynamic process of physical annealing of solids. In this process, a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration, i.e. its minimum lattice energy state, and is thereby free of crystal defects. If the cooling is sufficiently slow, the final configuration generates a solid with superior structural framing. SA uses this phenomenon in the search for a global optimum for a discrete optimization problem.

The key feature of the SA algorithm is that it provides a possibility to escape local optima by allowing hill-climbing moves, i.e. moves that worsen the objective function value. This possibility is determined by the temperature parameter that when it decreases, it reduces the possibility of a deteriorate move. In the start of the algorithm, the parameter is given a high value and allows thereby hill-climbing moves more frequently. This can be more easily understood by imagining a ball that bounces over mountains from valley to valley. When the temperature parameter is given a high value, the ball can bounce over any mountain, assuring itself access to whichever valley. But as the temperature decreases, the probability of bouncing over a certain mountain is lower, limiting the possible valleys to explore, and finally the ball will stop bouncing. In practice, two objective function values are compared; the one of a newly generated candidate solution to the one of the current solution. Improving solutions are always accepted, while deteriorating solutions are accepted by an established probability function, governed by the temperature parameter, which allows the algorithm to escape from local optima.

The Procedure:

SA always starts with an initial solution that has been established either randomly or by another heuristic technique. From this initial solution, a neighbour solution is generated, also either randomly or by using specific rules. To evaluate the neighbour solution, SA applies an acceptance probability that is based on the Metropolis acceptance criterion [12].

The neighbour solution (*w*') is always accepted as the new current solution (*w*) if its objective function value is the same or better than that of the current solution, i.e.

$$f(w') \leq f(w)$$

In the case where the objective function value of the neighbour solution is worse than the one of the current solution, i.e.

$$f(w') > f(w)$$

the neighbour solution is accepted as the new current solution with a probability given by:

$$p = e^{-\frac{\Delta}{T}} \qquad \text{(Eq. 6.1)}$$

where

$$\Delta = f(w') - f(w) \qquad \text{(Eq. 6.2)}$$

and *T* being the temperature parameter such that $T > 0$ [14]. This acceptance probability is the basic element of the search mechanism in SA, and if the temperature is decreased sufficiently slowly, through a cooling schedule, then the system can reach equilibrium.

To be able to execute a successful SA, the following elements should be provided:

- A well-designed representation of the solutions
- A neighbourhood generator that creates random changes in the current solution
- A way of evaluating the objective function value, f, of each solution
- A schedule for the cooling of the temperature parameter

The representation of the solution is dependent of the type of the problem, however, if it is constructed in a straightforward manner, both the generation of neighbours and the evaluation of the solutions will be facilitated. There are various ways of generating neighbours (now considering the scheduling problem), such as for example lot interchange, lot insert, lot merge, lot split, item interchange, item insert [27] and multi-exchange methods

[5]. However, regardless of the method, the generation of neighbours in SA always includes an element that is haphazard. It could be implemented in the selection of the item that will be inserted or in the selection of the positions of the two items that will be interchanged. This element of the SA algorithm is based on Monte Carlo methods that in general rely on repeated random sampling to compute their results [17]. Since there are often several haphazard elements in a SA algorithm, and every time it is executed, thousands of iterations are performed in the search for the optimal solution, it is also important that the random number generator used is of good quality.

A well-structured representation of the solution also facilitates the evaluation of each neighbour generated, i.e. the calculation of the objective function value (f) and the comparison with the one of the current solution. This is most important, above all for the general computational performance, since in most SA algorithms the part of the evaluation is the most computational intense activity [14]. This part also includes a haphazard element, when to determine whether or not to accept a neighbour solution with worse objective function value.

The last part of a well-designed SA algorithm that should be well defined, is the cooling schedule of the temperature parameter. In most SA algorithms, there are three elements that need to be established; the initial temperature T_0, a final temperature T_f (or another termination criterion) and a rule for decrementing the temperature. The initial temperature should adopt a value so that the probability of accepting neighbour solutions with worse objective function value than the current solution is high. As the temperature decreases with the iterations, this probability also decreases. There are different ways of establishing the rule of how the temperature should diminish, and the schedules are grouped into two classes: static and adaptive schedules [14]. A static schedule must be completely specified before the execution of the algorithm and two commonly used schedules are the linear and the exponential schedule:

$$T_i = T_{i-1} - T \qquad \text{(Eq. 6.3)}$$

and

$$T_i = \beta \cdot T_{i-1},\ i = 1,\dots,n,\dots,\infty,\ \beta<1 \qquad \text{(Eq. 6.4)}$$

An adaptive schedule on the other hand will adjust the temperature's rate of decrease from information provided during the execution of the algorithm. Often these schedules do not decrease the temperature until after a certain criterion is met, e.g. a number of iterations without changing the current solution. No matter what type of schedule is used, they are often heuristic and seek to balance a moderate running time with SA's dependency on asymptotic behaviour.

As mentioned before, any SA algorithm needs a termination criterion. If it is part of the cooling schedule, a final temperature T_f is often determined and the iterations stop when this temperature is reached. More general stopping criterions are commonly used, for example running time, number of iterations, number of iterations without improving the current solution, etc.

Comparison with Other Metaheuristics:

SA is a robust technique and can be used in a number of various problems including highly nonlinear ones and more chaotic problems with a lot of constraints. The main advantage over local search algorithms in general is its ease to escape from local optima and its ability to approach the global optimum [14]. Furthermore, SA is easy to adapt to specific problems, and it is also easy to implement more elements to the model so that the search can be more intelligent and adaptable.

The criticism towards SA includes often the time/quality issue. The algorithm requires often a considerable computational time to be able to locate the optimal solution (or near-optimal). If the method is compared to Genetic Algorithms, GAs often converges towards a good solution faster than SAs. Research has been done combining the two principles, and the resulting method has been able to converge much faster towards good solutions [21].

Another criticism of SA is that the algorithm does not use any kind of memory while searching for the optimal solution [14]. For example, a Tabu Search algorithm can store information, allowing it to intensify or diversify the search in the solution space of the given problem. Nevertheless, the general Tabu Search algorithm does not have the converging qualities that SA is equipped with.

Like most metaheuristics, SA also needs some kind of adjustment before working well in a specific problem. Since there are often a couple of parameters that need to be determined before executing the algorithm, some kind of initial tests should be performed to monitor how the algorithm performs when altering these parameters.

7 PROBLEM STATEMENT

When giving details on the different parts of the stated scheduling problem, the definitions used for multicriteria scheduling are applied:

- **Modelling of the problem**. This permits determining the nature of the problem as well as the definition of the criteria to be taken into account.
- **Taking into account the criteria**. The way in which the criteria are taken into account, also called a module, is defined.
- **Scheduling**. An algorithm for solving the problem is presented, also called resolution module, which finally leads to the solution of the problem. This part is presented under Chapter 8 Development of the Resolution Procedure.

7.1 Modelling of the Problem

7.1.1 The Environment

The study considers scheduling of identical parallel machines subject to the minimization of the maximum completion time and the maximum tardiness expressed in a linear convex objective function. The maximum completion time or makespan is the date when the last job to be completed leaves the system. The maximum tardiness is indicated by the job that is completed with the longest delay relative its due date. Minimizing both criteria can help assuring a high utilization of the production system as well as a high level of service towards the client.

A set of *n* jobs is assigned, to one of the *m* identical parallel machines. Each job is processed in only one operation before its completion after which it leaves the system. Constraints, such as due dates for each job and setup times for the machines, are considered.

The hypotheses presented by Conway, Maxwell and Miller (1967) are commonly used when dealing with the job-shop problem and are also used here to define the stated problem [25]:

- Each machine is continuously available from moment $f \geq 0$ until T, T being a sufficiently large number.

- The job routs never converge or diverge through assemblage or dismantling of jobs. Each operation follows another and is followed by other except the first and last operation of each job.
- Each operation can only be performed in one type of machine in the job-shop.
- Once started, an operation cannot be interrupted until it is completed.
- Two operations are not allowed to overlap each other.
- Each machine can only realize one operation at a time.
- The only active restriction in the shop is related to the machines.

No idle time in the machines between operations is allowed once the processing of jobs has commenced, i.e. once a machine completed the operation of a job it is prepared immediately for the operation of the next.

The resulting schedule will present to which of the machines the jobs are assigned, and also in which order in each machine the jobs assigned to that machine are processed.

7.1.2 Problem Parameters

The problem parameters dealt with, which for each instance (*I*) treated are given through the input data, will be presented in continuation. For the complete overview of the input data, an example is given in Appendix A.

Processing Time:

Since the machines are identical, the processing time depends only on the job itself in the machines. In any machine *i* the job *j* needs a specific amount of time to be processed, p_j:

Table 5.1. Processing time for each job in any of the machines.

p_j	**Job 1**	**…**	**Job *n***
Machine *i*	p_1	p_j	p_n

Due Dates:

For each job *j* a preferred instance when the job should be manufactured is established, d_j:

Table 5.2. Due dates for each job.

Job	**1**	**…**	***n***
d_j	d_1	d_j	d_n

Setup Times:

Every job needs some time to be prepared before starting its manufacturing in the machines, weather it is the first job to be processed in the machine or it is a job in between two others. This setup time is therefore said to be sequence dependent and is caused by the following activities:

- Stop the process.
- Cleaning of the machine, to remove contingent residues from the former job.
- Change of parts in the machine to fit the next job.
- Start-up of the process.

Since the machines are identical, the setup times depend only on the former job processed *j* and the next job *k*. The setup times between the jobs S_{jk} can be featured through a matrix as demonstrated in Table 5.3.

Table 5.3. Matrix with setup times between the jobs.

S_{jk}	**Job 1**	**Job *i***	**Job *n***
Status quo	S_{01}	S_{0i}	S_{0n}
Job 1	0	S_{1i}	S_{1n}
Job *i*	S_{i1}	0	S_{in}
Job *n*	S_{n1}	S_{ni}	0

An important observation to be made for sequence dependent setup times is that from the start the machines are not prepared to process none of the jobs, and hence a setup time is required depending on which job is set to be processed first. Another important point is that the setup time between job *j* and job *k*, S_{jk}, is never guaranteed to correspond to the setup time between job *k* and job *j*, S_{kj}.

7.1.3 Variables

The objective is to find a solution, applying different procedures, by creating a schedule, which allows optimizing the bicriteria objective function. To be able to develop an algorithm that seeks this objective it is necessary to establish and define variables and constraints.

• Completion time: The instant at which the job *j* is completed:	c_j
• Lateness:	L_j
• Tardiness:	T_j

The lateness indicates whether the job *j* has exceeded its due date or not when completed and is given by:

$$L_j = c_j - d_j \qquad \text{(Eq. 7.1)}$$

The tardiness indicates the time with which the job *j* is completed after its due date and is given by:

$$T_j = \max(L_j, 0) \qquad \text{(Eq. 7.2)}$$

7.1.4 Constraints

• The due date for each job *j*:	d_j
• Setup time between two jobs *j* and *k*:	S_{jk}

Furthermore, preemptions are not allowed, i.e. once a machine has started processing a job it is not possible to interrupt the process until the job is completed. Operations are not allowed to overlap each other and each machine can only realize one operation at a time.

7.2 Taking Into Account the Criteria: The Objective Function

The problem with identical parallel machines could be approached in many different ways, creating a schedule to suite for example a specific production plan. The different requirements of the production are translated into a measurable criterion that is included in the objective function.

The stated problem is approached through a bicriteria. The two objectives are expressed through a linear convex objective function and the resolution procedure has the intention of minimizing both the maximum completion time and the maximum tardiness:

$$[\min]F_\ell(c_{\max},T_{\max}) = \alpha \cdot [\max]c_j + \beta \cdot [\max]T_j \qquad \text{(Eq. 7.3)}$$

where $\alpha + \beta = 1$

The two criteria are often considered in the real-world industry since they are good indicators of the quality of the production in terms of overall performance and service offered towards the client.

- $[\min]c_{\max}$ or the minimization of the maximum completion time (makespan), tends to force a minimization of the setup times resulting in the balancing of the work load over the machines and gives therefore a high utilization of the system. The maximum completion time is given by the job last in leaving the system:
$$c_{\max} = [\max]c_j$$
- $[\min]T_{\max}$ is more direct related to the service given to the client since it minimizes the worst performance of the system in terms of consignment. Keeping *T_{max}* low means that the general service towards the client is guaranteed. The maximum tardiness is indicated by the job that is completed with the longest delay relative its due date:
$$T_{\max} = [\max]T_j = [\max](c_j - d_j, 0)$$

Finally, the complete notation of the stated problem according to the notation scheme of Graham et al. (1979) [20]:

$$P_m / d_j, S_{jk} / F_\ell(c_{\max}, T_{\max})$$

8 DEVELOPMENT OF THE RESOLUTION PROCEDURE

The resolution procedure proposed is the Simulated Annealing (SA) heuristic. The general theory of the procedure has been explained in Chapter 6.3.5.

Following the classifications of resolution procedures for multicriteria scheduling, the type of resolution procedure proposed is that of a priori method, i.e. the importance of the criteria α is set in the objective function and the procedure returns one unique solution in form of a schedule after a number of iterations.

The SA has been proven to work well with this particular scheduling problem, dealing with identical parallel machines, as proven by Lee et al. (2006) [8]. SA has also been proven to work well with bi-criteria objective functions, as it has been demonstrated by Ruiz-Torres et al. (1997) [26].

The design of the procedure was developed so that it consists of two phases in its approach on solving the stated scheduling problem:

1. **Phase 1: Creation of an initial solution**: In this phase the data is treated and the procedure presents one initial solution, in the form of a schedule, based on certain rules.
2. **Phase 2: Improvement Metaheuristic**: The main phase then includes the SA algorithm where neighbour solutions, based on the initial one, are generated and evaluated to find a good solution when searching the solution space. In each iteration, only one neighbour is generated and evaluated through its objective function against the current solution. This phase ends when a termination criterion is met. The best solution found during this phase is then presented in the form of a schedule.

A general overview of the procedure is shown in the Figure 8.1 below:

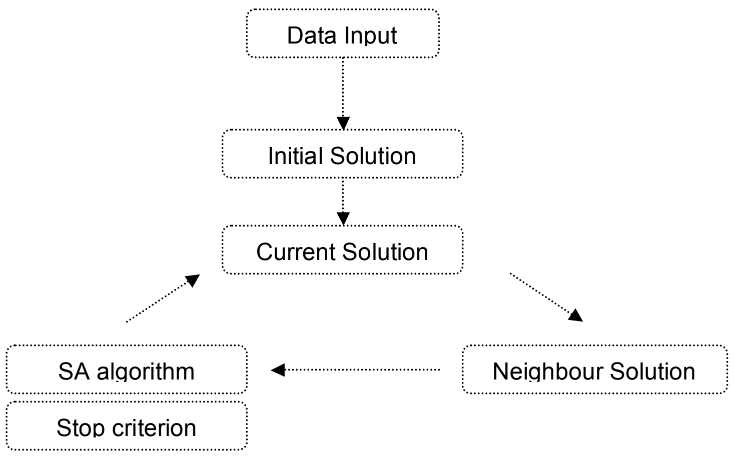

Figure 8.1. General overview of the heuristic procedure.

8.1 Phase 1: Creating an Initial Solution

In the design of the algorithm, four methods of creating an initial solution have been developed. Since the procedure is a metaheuristic, its characteristics are of such kind that an initial solution normally is generated by another algorithm, normally a deterministic one, and used as a starting point in the search for a better one. In some literature it is claimed that the characteristics of the metaheuristic are such that in general they do not require a good starting solution to be able to find a near-optimal one. Therefore only one method to create an initial solution is used [8]. However, many papers on the subject also claim the opposite. Different methods are then developed to create a good initial solution [26]. Since both cases exist and none has been proven completely right, several methods to create an initial solution are developed in this study and tested before drawing any conclusions.

The procedure of creating an initial solution is divided into two phases:

- Order the jobs in a list according to a priority rule.
- Assign the jobs in that order to the machine where the completion time for that job is minimized, considering the completion time for the anterior job and adding both the setup time required and the processing time for the assigned job.

In all cases, c_{max}, T_{max} and the objective function value, F_ℓ, of the initial solution is calculated after it has been created.

8.1.1 SPT

The first method organizes the jobs in a list following the SPT rule, so that the job with the shortest processing time is first in the list and the second job in the list is that with the second shortest processing time, etc. Next, each job is assigned, following the order of the list, to that machine where the completion time for the job is minimized, considering both the processing time and the setup time for that job. For an example, see Figure 8.2.

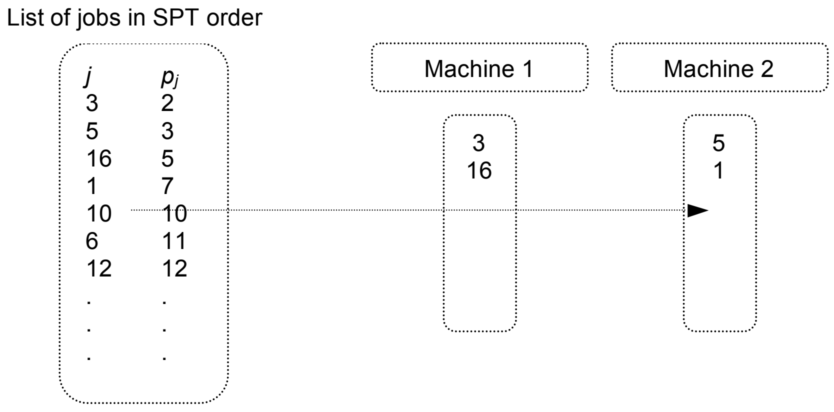

Figure 8.2. Assignment of jobs to machines using the SPT priority order.

8.1.2 EDD

The next method organizes the jobs in a list following the EDD rule, so that the jobs with the earliest due date is first in the list and the second job in the list is that with the second earliest due date, etc. Next, each job is assigned, following the order of the list, to that machine where the completion time for the job is minimized, considering both the processing time and the setup time for that job. For an example, see Figure 8.3.

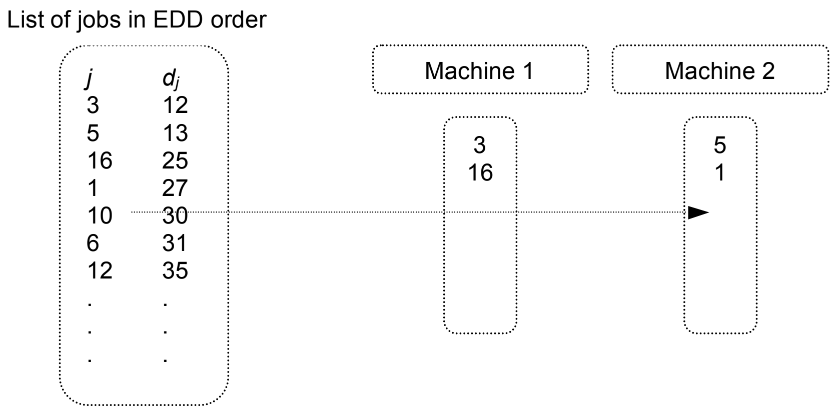

Figure 8.3. Assignment of jobs to machines using the EDD priority order.

Applying this rule prioritizes the completion of the most urgent jobs and has been shown to minimize the maximum lateness of all jobs, L_{max}, and therefore it also has a direct impact on the maximum tardiness, T_{max}.

8.1.3 Weighted SPT & EDD

Another rule applied when creating the initial solution is a convex linear function of the two earlier mentioned priority rules where they each are given a weight to express their importance. This rule is applied to discover if the merged function of the two rules could have a better impact on the initial solution than any of the two extremes. Since the EDD rule has been shown in the literature to have an impact on the T_{max} criteria, this rule is given more weight in the merged function:

$$f_m = 0.2 \cdot p_j + 0.8 \cdot d_j \qquad \text{(Eq. 8.1)}$$

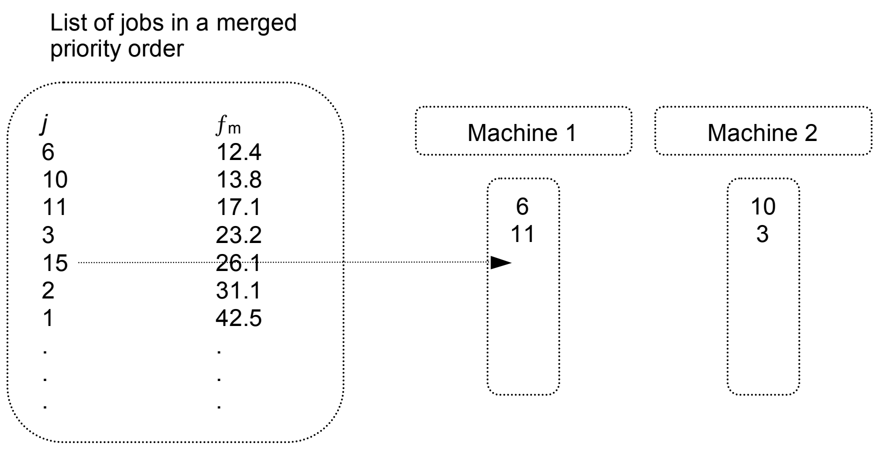

Figure 8.4. Assignment of jobs to machines using the merged priority order.

8.1.4 Random

Finally, the last method used when creating an initial solution is based on total randomness. The jobs are randomly listed, and are assigned in that random order to the machines one by one creating a random, but balanced schedule, see Figure 8.5. This method of creating the initial solution is the most basic one and is used as a reference to be able to compare the efficiency of the other methods.

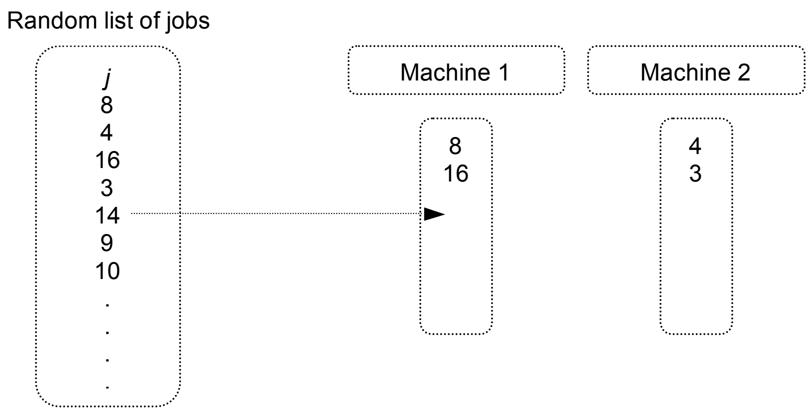

Figure 8.5. Assignment of jobs to machines using a random order.

8.2 Phase 2: Improvement Phase

8.2.1 The Simulated Annealing Algorithm

Once the initial solution has been created, the procedure starts to search for better ones by generating neighbour solutions to explore the solution space. In each iteration, only one neighbour is generated from the current solution and its c_{max}, T_{max} and F_ℓ is calculated. The algorithm then has to decide weather to move on searching for better solutions in that direction or change direction and explore another part of the solution space. The decision lies in accepting the neighbour solution generated as the new current solution or not. Two cases can occur:

- The neighbour solution generated <u>is accepted</u> as the new current solution and the sequence as well as c_{max}, T_{max} and F_ℓ are stored for the new current solution. In the next iteration, a neighbour solution is generated from this new current solution and the SA algorithm once again decides weather to update the current solution or not. Furthermore, if the neighbour solution is better than the, up to that moment, best solution found, that solution is updated in terms of sequence, c_{max}, T_{max} and F_ℓ.

- The neighbour solution generated <u>is not accepted</u> as the new current solution. The algorithm continues to search the solution space in another direction generating a new neighbour solution from the old current solution, etc.

The decision element in the SA algorithm is designed with three independent rules. First, an evaluation is realized considering the objective function value of the neighbour solution in comparison with that of the current solution:

$$\Delta = F_\ell(neighbour) - F_\ell(current) \qquad \text{(Eq. 8.2)}$$

Depending on the objective function value of the neighbour and the current solution, three situations can occur, each with its own rule of decision making:

1. $\Delta < 0$. This means that the neighbour solution is better than the current solution and it is thereby accepted automatically as the new current solution. The sequence as well as the values of c_{max}, T_{max} and F_ℓ are updated.

2. $\Delta = 0$. The neighbour solution has the exact same objective function value as the current solution. To give more dynamics to the algorithm, in this case of draw the neighbour solution is accepted as the new current solution with a randomly generated probability. All cases of draw are therefore not accepted, but they are rather given a probability of being accepted.

   The acceptance of ties is controlled by a variable, *same*, that counts the ties. When a certain number of ties have been reached, accepted or not, the ties are no longer given any probability of being accepted. This implementation is created to control the running time.

3. $\Delta > 0$. The neighbour solution is worse than the current solution. The acceptance is controlled by a dynamic probability given by:

   $$p = e^{-\frac{\Delta}{T}} \quad \text{(Eq. 8.3)}$$

   If Δ is a large number, i.e. the neighbour solution is much worse than the current solution, the probability of accepting it is low. If Δ is smaller, the probability of accepting the slightly worse neighbour solution is higher.

   Another parameter is *T* or the temperature, which also affects the probability. In the beginning *T* is a large number but it decreases, in case of accepting a neighbour solution as the new current solution (in any of the three ways), as the search proceeds following a cooling schedule:

   $$T_i = T_{i-1} \cdot \beta;\ 0 < \beta < 1 \quad \text{(Eq. 8.4)}$$

   Since the value of *T* decreases when a neighbour solution is accepted, and the probability is given by an exponential function, if the algorithm proceeds during too many iterations, *T* approaches zero and the probability becomes infinite. To avoid this scenario, the value of *T* is adjusted by an implemented security. When *T* hits a lower limit, this security resets the value of *T* to a somewhat higher value from which *T* then can decrease anew.

In practice, when starting the SA algorithm, the probability of accepting even much worse neighbour solutions is high, allowing the algorithm to escape from local optima. As neighbour solutions are accepted, and the temperature decreases, this possibility also diminishes, making it less likely to accept much worse neighbour solutions and the algorithm by some means converges. Yet, the algorithm could accept worse solutions due to this probabilistic element even in the latter part of the search. As can be seen in Figure 8.6, the characteristics of the SA algorithm (the example is from the SAS algorithm) when the value of the objective function of the current solution is monitored over time. The tendency is slightly convergence, and the peaks demonstrate the situations when a much worse neighbour solution is accepted to escape from local optima. As can be seen, the peaks are also more frequent in the beginning, but as more neighbour solutions are accepted and *T* decreases, the peaks become less existent.

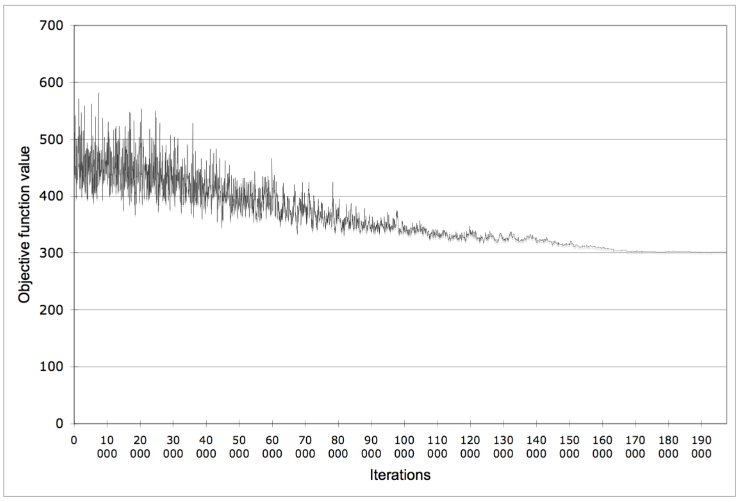

Figure 8.6. Graph of the SAS algorithm, monitoring the F_ℓ of the current solution.

8.2.2 Stop Criteria

In the design of the procedure, two stop criteria are used:

- *terminate*: Number of iterations without improvement of the current solution.
- *itmax*: Maximum number of iterations.

Basically, each time that the neighbour solution generated is not better than the current solution in terms of F_ℓ, i.e. when $\Delta \geq 0$, a variable increases one step. When the variable reaches a pre-established limit, it makes the algorithm stop. If during this process, the procedure should find a neighbour solution that in fact has a better F_ℓ than the current solution, i.e. $\Delta < 0$, the variable is anew set to zero, and the counting starts all over again.

If the procedure during the search process is able to find better neighbour solutions compared to the current ones over and over again, the stopping criterion could have difficulties of being reached. In that case, to be able to find a solution within a feasible amount of time, the number of iterations is then set to limit the search. When the procedure has reached a pre-established number of iterations, but the first stopping criterion has not been reached, the algorithm is also stopped.

In either case, when the procedure stops, the best solution obtained during the search is presented with its schedule together with the values of its c_{max}, T_{max} and F_ℓ.

8.3 Neighbourhood Generation

In the design of the procedure, three ways of generating neighbour solutions are used:

- *Swap* or *SAS* (Simulated Annealing with Swap)
- *Insert* or *SAI* (Simulated Annealing with Insert)
- A version where both of the methods are used, one after the other or *SASI* (Simulated Annealing with Swap then Insert)

The first two methods are commonly used with scheduling problem for their easy implementation and are proven to function well with the SA algorithm contributing to the achievement of good results like demonstrated by Kim et al. (2002) [27] and Ruiz-Torres et al. (1997) [26].

Since the first two methods search for solutions in the solution space in different ways, each one might not be able to search every part. The third method is therefore implemented to be able to do a more meticulous search of the solution space as it combines the two methods.

After each generation of a neighbour, independently of the method used, the c_{max}, T_{max} and F_ℓ are all calculated for that solution. The solution sequence is also stored.

8.3.1 SAS

The first method, *SAS*, selects two of the jobs randomly without considering their positions in the sequence or in the machines and interchanges their position as can be seen in Figure 8.7. The jobs swapped could be positioned in the same machine, sequenced after each other in that sequence or be positioned in different machines.

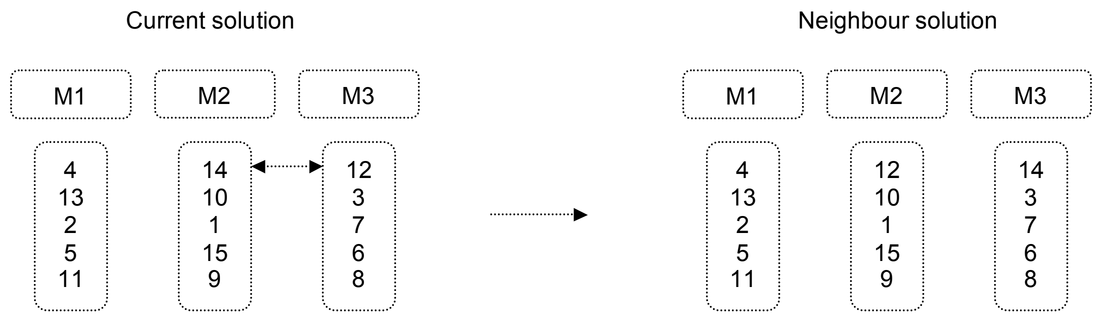

Figure 8.7. The generation of a neighbour solution using the algorithm *SAS*.

8.3.2 SAI

The second method, SAI, selects first one job randomly and then selects a machine in the same way. The machine selected could be the same machine in which the selected job currently is positioned. Somewhere in the sequence in the selected machine, a position is then randomly chosen where the job is inserted. The selected job could be inserted first or last in a sequence or in between two other jobs, depending on the position that is randomly chosen. For an example, see Figure 4.5.

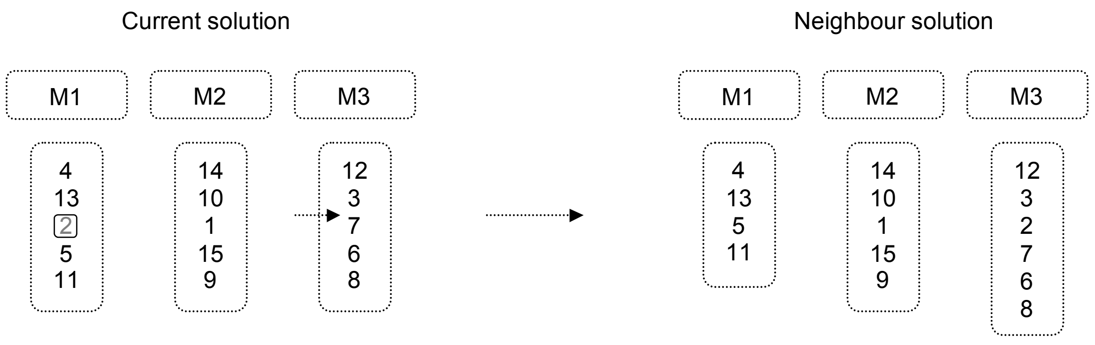

Figure 8.8. The generation of a neighbour solution using the algorithm *SAI*.

8.3.3 SASI

The third procedure used when searching the solution space uses first the SAS algorithm and then, when a criterion, *change*, is met, it changes to SAI and proceeds until the stop criterion *terminate* is met. The criterion *change* is constructed in the same way as the stop criterion *terminate*, i.e. it counts the number of times the algorithm is unable to find a neighbour solution that is improving the F_ℓ compared to the current solution. If $\Delta < 0$, *change* is reset to 0. When having reached its limit, it makes the algorithm switch to the other neighbour generating method. This way, if one method makes the algorithm unable to escape from local optima, the other might be able to do so as they do not search the solution space in the same way. When switching from one method to the other, the value of *T* is reset to its initial value and *terminate* and *same* are reset to 0 to give the procedures equal conditions and more probability of escaping from local optima. In Figure 8.9 an example is given showing the behaviour of the SASI algorithm.

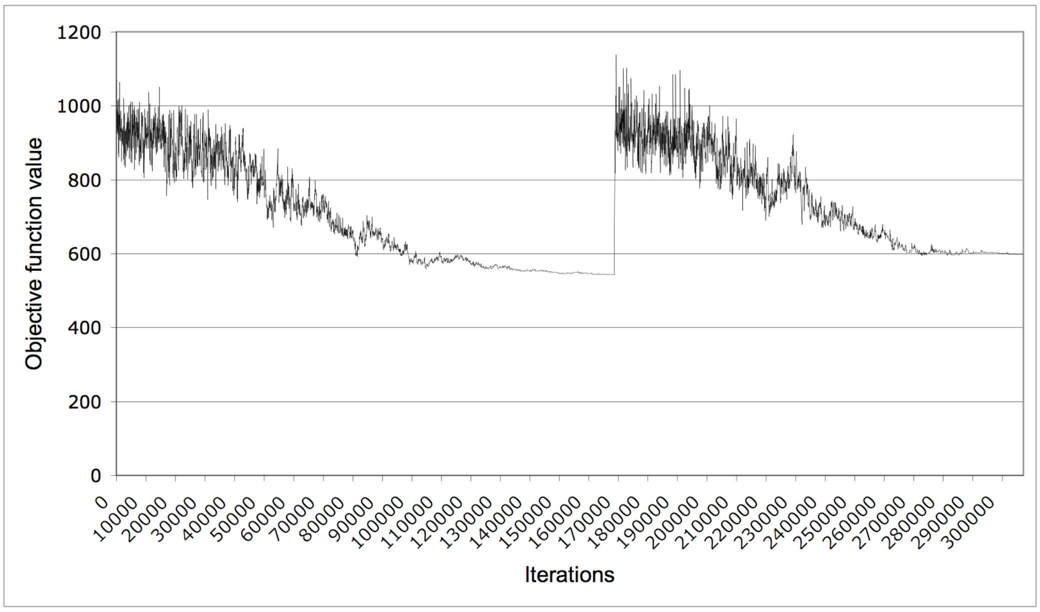

Figure 8.9. Graph of the SASI algorithm, monitoring the F_ℓ of the current solution.

8.4 Procedure Flow Diagram

A complete overview of the designed SA algorithm is shown in Figure 8.10 where the whole chain of decision-making can be followed. For explanations to abbreviations, see Chapter 1 Abbreviations and Symbols.

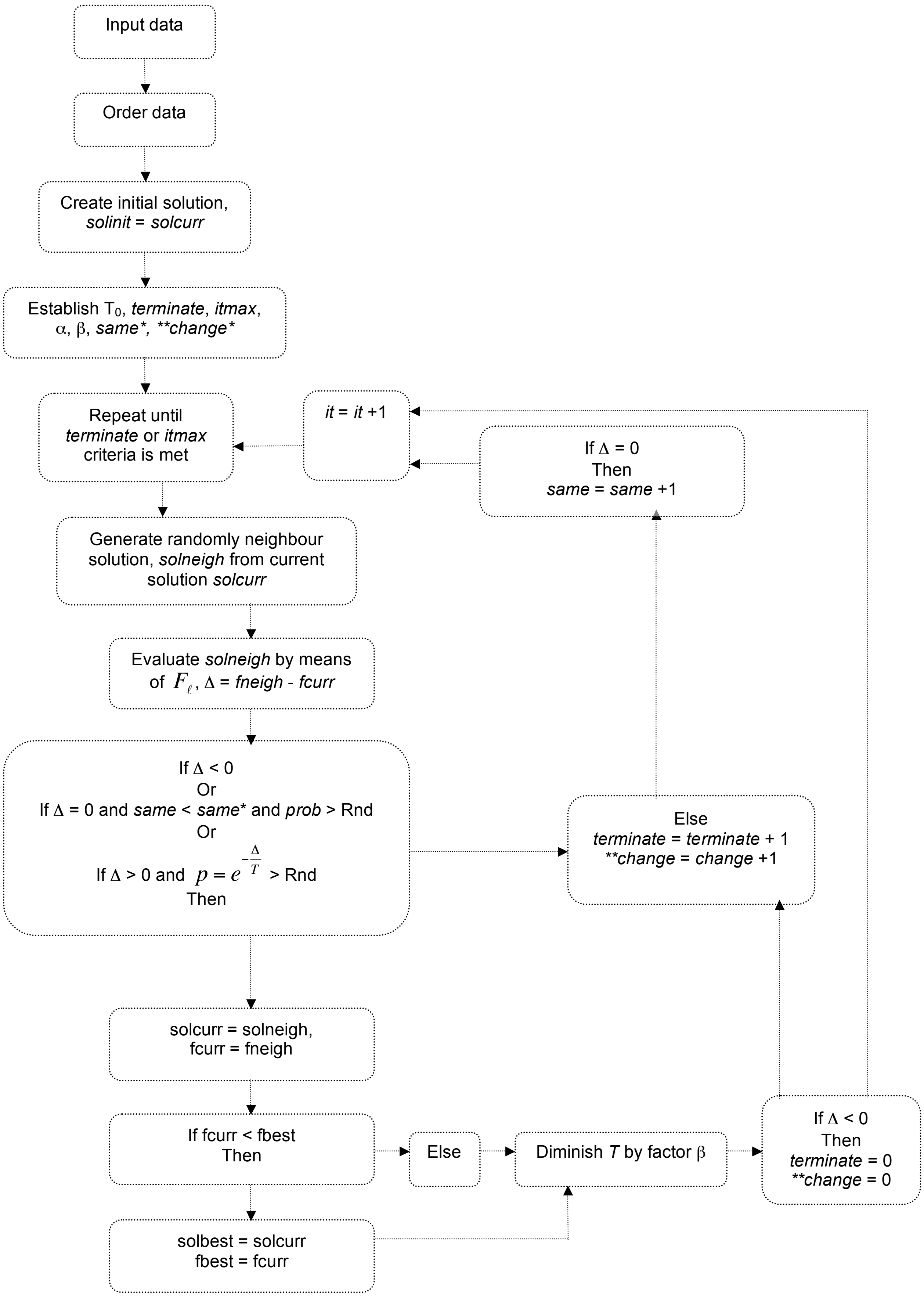
Input data
Order data
Create initial solution, *solinit* = *solcurr*
Establish T_0, *terminate*, *itmax*, α, β, *same*, **change**
Repeat until *terminate* or *itmax* criteria is met
it = *it* +1
If Δ = 0 Then *same* = *same* +1
Generate randomly neighbour solution, *solneigh* from current solution *solcurr*
Evaluate *solneigh* by means of F_ℓ, Δ = *fneigh* - *fcurr*
If $\Delta < 0$ Or If $\Delta = 0$ and *same* < *same** and *prob* > Rnd Or If $\Delta > 0$ and $p = e^{-\frac{\Delta}{T}}$ > Rnd Then
Else *terminate* = *terminate* + 1 ***change* = *change* +1
solcurr = solneigh, fcurr = fneigh
If fcurr < fbest Then
Else
Diminish *T* by factor β
If $\Delta < 0$ Then *terminate* = 0 ***change* = 0
solbest = solcurr fbest = fcurr

Figure 8.10. Overview of the designed SA algorithm. **When using SASI

8.5 Summary

To study the stated problem with the bicriteria objective function a resolution procedure has been designed that consists of two mayor phases, the phase where an initial solution is created and the phase where a search is performed to find a better solution. For the first phase, four procedures where developed and for the second phase there are three variants of the SA algorithm implemented, as can be seen in Figure 8.11.

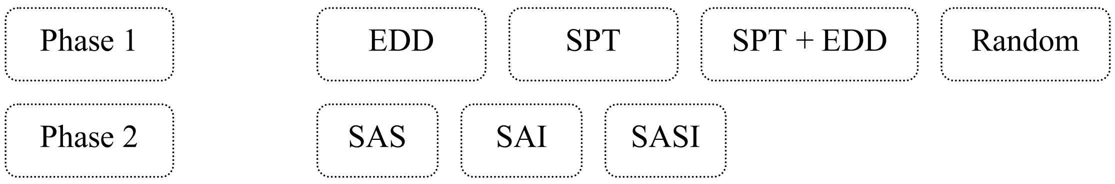

Figure 8.11. Overview over the different procedures proposed of the two phases of the heuristic.

The combinations of the procedures results in a set of 12 algorithms to tackle the stated problem, as can be seen in Table 8.1.

Table 8.1. The complete set of algorithms developed to tackle the stated problem.

	EDD	**SPT**	**EDD + SPT**	**Random**
SAS	ESAS	SSAS	ESSAS	RSAS
SAI	ESAI	SSAI	ESSAI	RSAI
SASI	ESASI	SSASI	ESSASI	RSASI

9 COMPUTATIONAL EXPERIMENTATION

9.1 Overview

The computational experimentation has as objective to study the behaviour of the proposed resolution procedure while dealing with the stated problem. To do so, some hypotheses has been stated:

- **H1**: Any of the proposed procedures for creating an initial solution is better than the others in terms of the value of the objective function of the initial solution.
- **H2**: The proposed procedure, the Simulated Annealing, needs a good initial solution to perform better while dealing with the stated problem considering the bicriteria.
- **H3**: Any of the proposed methods for generating neighbour solutions leads to better results than the others in terms of the value of the objective function of the proposed solution schedule.

To be able to realize the study a Simulated Annealing algorithm has been implemented using the Visual Basic© 6.0. In Appendix C, the code used for the implementation of the SA algorithm is presented. All the experiments have been performed on a Dell Optiplex GX620 with a Pentium D 3 GHz processor and 512 MB RAM.

9.1.1 Input Data

Before initializing the first phase of the resolution procedure, where an initial solution is created, the input data is read for the specific instance. The data is stored in vectors and matrixes depending on the amount of data. The input required by the algorithm and given for each instance is:

- Situation: number of machines *m* and number of jobs *n.*
- Matrix with setup times for each job in any machine, S_{jk}.
- Vector with the processing time for each job, p_j.
- Vector with the due date for each job, d_j.

To be able to realize various analyses and investigate the stated hypotheses, a collection of test problems was generated *ad hoc* following the structure given by Potts and Van Wassenhove (1982) [4]. The data presented was generated so that five different kinds of configurations of jobs and machines could be analyzed; 15, 20 or 50 jobs scheduled on 2, 3 or 4 machines as demonstrated in Table 9.1.

Table 9.1. The different job and machine configurations presented through the input data provided.

Jobs	Machines
15	2
20	2
20	3
50	3
50	4

The collection of data consisted of 240 sets of instances, D_{Π}, with 10 instances, I, in each. Among the 2400 instances in total there were not 2 alike. The instances provided were divided equally among the five different configurations, 480 instances each. In each set of instances, the 10 instances had distinct p_j and d_j, but all 10 shared the same data for S_{jk}. The setup times and the due dates had been created for each job in accordance with different parameters, also demonstrated in each set of instances. The creation of the setup times was controlled by a parameter λ and the due dates by two parameters, α and β. An example of a set of instances with input data is given in Appendix A.

9.1.2 Output Data

The results given by the heuristic are:

- The sequence of the best solution found during the search.
- The F_ℓ for the initial solution.
- The values of c_{max}, T_{max} and F_ℓ for the best solution found.
- In case of using SASI (see Chapter 8.3.3), F_ℓ for the best solution found by each algorithm.
- Number of iterations.
- Running time.

Additional data is withdrawn for each instance and in Table 9.2 all data withdrawn by the heuristic for each instance is demonstrated in order. In Appendix B two examples are given for the data presented by the heuristic together with the sequences of the initial and best solution.

Table 9.2. The output data given by the heuristic for each instance.

Index	Information
n	Number of jobs
m	Number of machines
ex	The number of the instance in a given set of instances
p1	δ
p2	λ
p3	α
p4	β
p5	Number of instances in the set of instances
finit	F_ℓ of the initial solution
fbesthalf	F_ℓ of the best solution found of the first algorithm when using SASI
fbest	F_ℓ of the best solution found (of the second algorithm when using SASI)
$c_{maxbest}$	c_{max} of the best solution found
$T_{maxbest}$	T_{max} of the best solution found
Time	Running time
it	Number of iterations
filepath	Information about the filename

9.1.3 Processing of Output Data: The Performance Index I_e

When studying the performance of the different procedures proposed, it was necessary to introduce an index that could represent the performance of the algorithm:

$$I_e = \frac{F_\ell - [\min]F_\ell}{[\min]F_\ell} \qquad \text{(Eq. 9.1)}$$

For each instance, the best F_ℓ presented by each algorithm, was compared to the $[\min]F_\ell$, i.e. the best F_ℓ among all the versions of the algorithms. In the case when the instances were processed a couple of repetitive times, the F_ℓ represents the average obtained for each algorithm (mean value). The resulting index I_e for each algorithm represents how much worse that algorithm is compared to the best result for that specific instance, and is expressed by a percentage. Since there were many instances, an average (mean) of the index I_e was calculated to be able to present a general performance of each procedure. This performance index I_e was used in all experiments and in all comparisons between procedures.

9.1.4 Experimental Strategy

The experimental approach to test the hypotheses stated is the one demonstrated in Figure 9.1. and explained, step by step in the following subchapters.

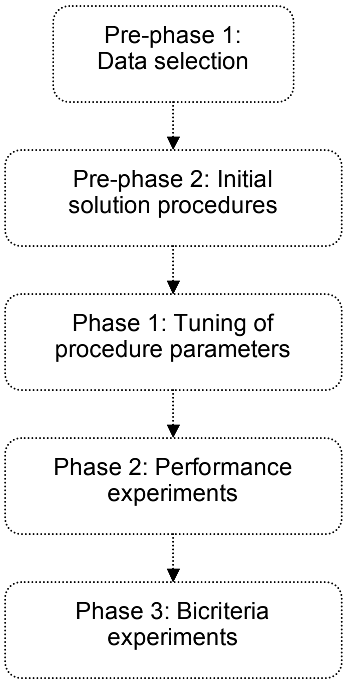
Pre-phase 1: Data selection
Pre-phase 2: Initial solution procedures
Phase 1: Tuning of procedure parameters
Phase 2: Performance experiments
Phase 3: Bicriteria experiments

Figure 9.1. Experimental strategy.

When realizing all the experiments, except in phase 3, the α in the objective function was set to 0.4, and automatically β was given the value of 0.6:

$$[\min]F_{\ell}(c_{\max},T_{\max}) = 0.4\cdot[\max]c_j + 0.6\cdot[\max]T_j \qquad \text{(Eq. 9.2)}$$
$$\alpha+\beta = 0.4+0.6 = 1$$

This way the criteria were balanced in the objective function giving slightly more importance to the T_{max} criterion. This configuration resulted successful in practice when realizing the experiments, creating schedules where both of the criteria were considered and actively influencing the objective function value.

The other parameters that were set when necessary were:

- β = 0.999 (temperature schedule)
- *same* = 1000
- *change* = 5000

The parameters were set regarding the running time, allowing the procedure to terminate within a feasible amount of time. *change* was set to 5000 giving the exact same opportunity to SAI as SAS when using SASI.

9.2 Pre-phase 1: Data Selection

9.2.1 Purpose

Since the criteria of the stated problem were expressed through a bicriteria linear convex function, this pre-phase was realized to discover if any of the large amount of data was more adequate than other when applying the proposed procedure. When implementing the heuristic it is important that both of the criteria are active in the objective function and that one of them does not adopt values that can be disregarded.

The experiment was realized applying three procedures to be able to analyze and perhaps find a pattern among the created input data that better suited the proposed approach to the problem:

- RSAS
- RSAI
- RSASI

The total collection of data D_{Π}, i.e. all 2400 instances were analyzed by each procedure.

9.2.2 Result & Conclusions

It was discovered that for many sets of data the T_{max} criterion did not have any relevance in the objective function, rather adopting the value 0, giving all the importance to the c_{max} criterion. Since the objective was to study the problem were both the criteria were active in the objective function, this type of data was discarded. It seemed that all data created with the parameter α, which controls the due dates, set to 0.8 was better adapted to he stated problem and the proposed procedure. In Appendix D, Pre-phase 1, examples of results are given to show this tendency.

When discarding the rest of the data, using only the data created with the parameter α = 0.8, a total of 600 instances were left as the data collection D_{Π}, equally distributed among the five configurations of jobs and machines, i.e. 120 instances for each configuration.

9.3 Pre-phase 2: Initial Solution Procedures

9.3.1 Purpose

The objective of this pre-phase was to deal with the hypothesis earlier stated:

- **H1**: Any of the proposed procedures for creating an initial solution is better than the others in terms of the value of the objective function of the initial solution.

The four procedures to create an initial solution, EDD, SPT, EDD + SPT and Random were used to realize the experiments. The whole collection of data D_{Π}, now consisting of 600 instances, was processed and for each instance an initial solution and its value of F_{ℓ} was presented by each procedure. To be able to compare their performance, the index I_e was used. For each procedure, an average I_e (mean) was then calculated for each of the job configurations 15, 20 and 50 jobs as well as the overall average (mean). The variance of the I_e was also calculated to see if the performance fluctuated much among the instances.

9.3.2 Result & Conclusions

The result of the experiment is shown in table 9.3. For examples of the results, see Appendix D, Pre-Phase 2.

Table 9.3. Comparison of the performance of the procedures for creating an initial solution.

	EDD		**SPT**		**EDD + SPT**		**Random**	
Configuration	I_e	σ^2	I_e	σ^2	I_e	σ^2	I_e	σ^2
15	**1,0 %**	0,0003	29,1 %	0,0274	2,2 %	0,0011	36,5 %	0,0369
20	**1,1 %**	0,0003	32,1 %	0,0326	1,9 %	0,0009	42,9 %	0,0521
50	**0,5 %**	0,0001	46,3 %	0,0446	1,7 %	0,0004	56,8 %	0,0557
Overall	**0,9 %**	0,0003	37,2 %	0,0419	1,9 %	0,0007	47,2 %	0,0573

The results show that for the bicriteria objective function with c_{max} and T_{max}, EDD is the preferable procedure when creating an initial solution, outperforming the others markedly. One can notice that the merged priority rule also presents good results, but yet they are slightly worse than those presented by the EDD rule. This is due to the influence of the SPT rule in the merged priority rule.

To acknowledge the hypothesis stated it is possible to say that the EDD rule is better than the others when creating an initial solution. To be able to test the other hypotheses stated, e.g. if the Simulated Annealing needs a good initial solution to perform better while dealing with the stated problem considering the bicriteria, the SPT rule and the merged priority rule were discarded. The rest of the experimentation was performed only with the EDD and Random procedures, having now six complete procedures to work with as indicated in Table 9.4.

Table 9.4. The set of procedures under consideration after the realization of the pre-phases.

	EDD	**Random**
SAS	ESAS	RSAS
SAI	ESAI	RSAI
SASI	ESASI	RSASI

9.4 Phase 1: Tuning of Algorithm Parameters

9.4.1 Purpose

Before realizing the main experiments to measure the performance of the proposed procedures, it was necessary to consider the effect of some of the algorithm parameters implemented to tune each procedure and to make them somehow perform better.

A decision was made to further investigate the effect on the performance of the proposed heuristic of two key parameters: the stop criterion (*terminate*) and the initial temperature (T_0). A series of experiments were conducted in accordance with the factorial design 2^k, where k = 2. The effect of the parameters on F_ℓ was analyzed through the performance index I_e for each instance processed, having the value of the parameters set in two levels. The advantage of using this type of experimental strategy is that it is sequential and hence all experiments are not realized at once, but rather in phases. This way it is easier to avoid false conclusions if the effects of the parameters, if any, do not have a lineal tendency.

The objective of the experiments was to discover if the parameters had any significant effect on the performance of the SA algorithm and the procedures to generate neighbour solutions. The experiments were realized with the following procedures and configurations:

- Procedures:
  - RSAS
  - RSAI
- Configurations:
  - 20 jobs
  - 50 jobs

The procedures were selected because the effect was to be analyzed without the possible influence of a procedure that could create a good initial solution. Random was therefore chosen as procedure to create the initial solutions and furthermore SASI was thought to be under the same effect of the parameters, if any, as SAS and SAI. The procedure was therefore not analyzed separately. Regarding the configurations, the situation with 15 jobs was not analyzed since it was thought that there was no significant difference between that configuration and the one with 20 jobs.

The levels for the parameters, low and high, where established based on observations during some initial experiments conducted on a smaller subgroup of instances. They were established focusing on controlling the running time, so that the procedures could return a good solution within a feasible amount of time. Within this time, a range of the parameters, represented by the two levels, was analyzed to be able to improve the performance of the procedures. The same levels were established for all configurations and both the procedures analyzed, as demonstrated in Table 9.5.

Table 9.5. Levels of the parameters for RSAS and RSAI.

Parameter	Low	High
T_0	100	500
terminate	5000	10000

Four experiments each for the two procedures were therefore realized as demonstrated by Table 9.6 and 9.7. The experiments were realized in a random order to minimize any external influence. In each experiment the sub-collection, earlier explained, of 480 (240 with 20 jobs and 240 with 50 jobs) instances was processed and the performance index I_e was calculated.

Table 9.6. Configuration for the experiments for RSAS.

StdOrder	RunOrder	T_0	*terminate*
4	1	500	10000
1	2	100	5000
3	3	100	10000
2	4	500	5000

Table 9.7. Configuration for the experiments for RSAI.

StdOrder	RunOrder	T_0	terminate
1	1	100	5000
2	2	500	5000
4	3	500	10000
3	4	100	10000

9.4.2 Result & Conclusions

The results for the two procedures and the two configurations are demonstrated in Table 9.8, 9.9, 9.10 and 9.11. The performance index I_e as well as its variance and the average (mean) running time are presented. The index is an average (mean) of the 480 instances processed. Only by observing the results it is not possible to draw any conclusions of the contingent effects of the parameters on I_e. It was necessary to let the results be analysed by Minitab© in order to further investigate the possible effects. For examples of output data and calculations, see Appendix D, Phase 1: Tuning.

Table 9.8. Results for the experiments for RSAS, 20 jobs.

StdOrder	RunOrder	T_0	*terminate*	I_e	σ^2	*Time (s)*
4	1	500	10000	0,81 %	0,000081	15.8
1	2	100	5000	0,97 %	0,000087	7.9
3	3	100	10000	0,92 %	0,000095	15.1
2	4	500	5000	0,82 %	0,000091	7.6

Table 9.9. Results for the experiments for RSAS, 50 jobs.

StdOrder	RunOrder	T_0	*terminate*	I_e	σ^2	*Time (s)*
4	1	500	10000	0,65 %	0,000320	43.1
1	2	100	5000	1,07 %	0,000089	24.1
3	3	100	10000	0,69 %	0,000060	43.8
2	4	500	5000	0,95 %	0,000084	24.6

Table 9.10. Results for the experiments for RSAI, 20 jobs.

StdOrder	RunOrder	T_0	*terminate*	I_e	σ^2	*Time (s)*
1	1	100	5000	2,24 %	0,000742	9.8
2	2	500	5000	2,61 %	0,000841	10.4
4	3	500	10000	2,05 %	0,000626	18.5
3	4	100	10000	2,29 %	0,000587	17.7

Table 9.11. Results for the experiments for RSAI, 50 jobs.

StdOrder	RunOrder	T_0	*terminate*	I_e	σ^2	*Time (s)*
1	1	100	5000	1,88 %	0,000456	33.0
2	2	500	5000	2,01 %	0,000618	33.2
4	3	500	10000	1,81 %	0,000471	58.6
3	4	100	10000	1,54 %	0,000348	55.9

The analysis of the results was realized with Minitab© and are presented as two plots:

- A normal probability plot of the effects on I_e (alpha = 0,05)
- The main effects on I_e

With these two plots, the effects that each parameter has on I_e will be discovered and also if the effects are statistically significant, i.e. if the effects shown in this experiment only are a coincidence or not considering a tolerance of 0,05. The results are presented in Figure 9.2-9.

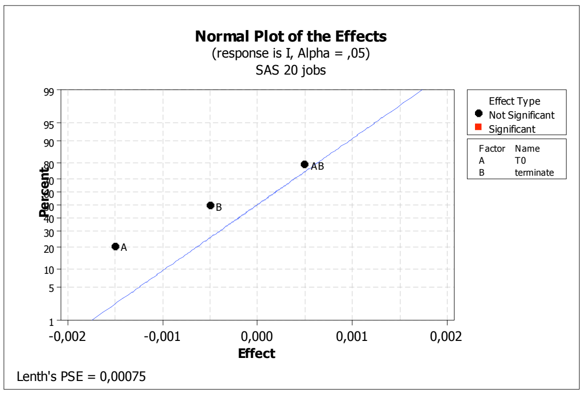

Figure 9.2. Normal probability plot of effects on I_e, *RSAS*, 20 jobs.

The Figure 9.2 shows that the individual effects as well as the combined effect of the two parameters are close to 0, and they are all considered being no significant given an alpha of 0.05.

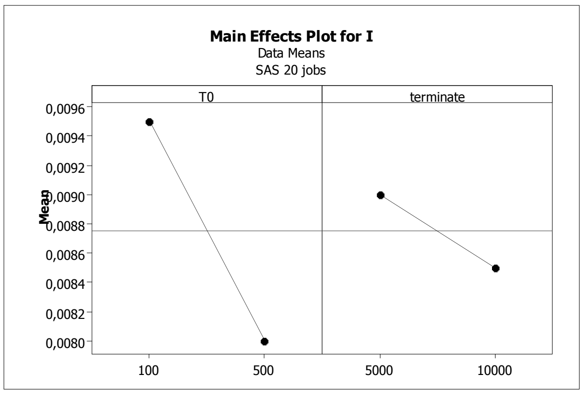

Figure 9.3. Plot of main effects on I_e, *RSAS*, 20 jobs.

It can be observed (in Figure 9.3) that when for example T_0 is changed from its lower value (100) to its higher (500), the mean of I_e diminishes from 0,0095 to 0,008. However, this effect cannot be proven to be significant and is in this case just a coincidence. The same trend is shown for the effect of *terminate*.

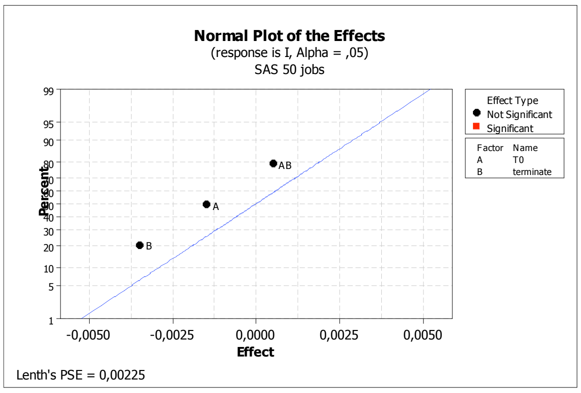

Figure 9.4. Normal probability plot of effects on I_e, *RSAS*, 50 jobs.

The Figure 9.4 shows that the individual effects as well as the combined effect of the two parameters are close to 0, and they are all considered being no significant given an alpha of 0.05.

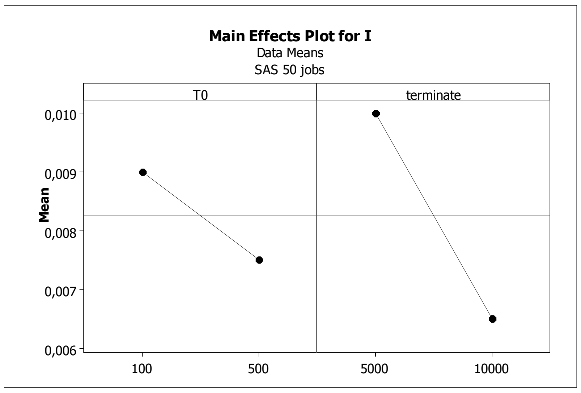

Figure 9.5. Plot of main effects on I_e, *RSAS*, 50 jobs.

It is shown (in Figure 9.5) that when for example *terminate* is changed from its lower value (5000) to its higher (10000), the mean of I_e diminishes from 0,01 to 0,0065. However, this effect cannot be proven to be significant and is in this case just a coincidence. The same trend is shown for the effect of T_0.

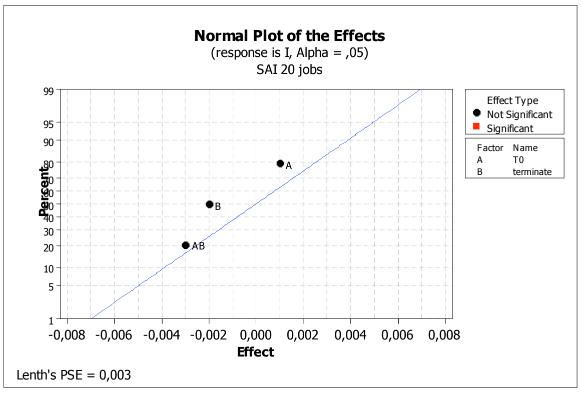

Figure 9.6. Normal probability plot of effects on I_e, *RSAI*, 20 jobs.

The Figure 9.6 shows that the individual effects as well as the combined effect of the two parameters are close to 0, and they are all considered being no significant given an alpha of 0.05.

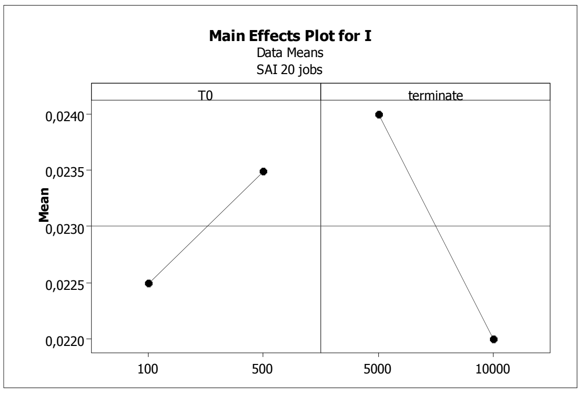

Figure 9.7. Plot of main effects on I_e, *RSAI*, 20 jobs.

The Figure 9.7 shows that when for example *terminate* is changed from its lower value (5000) to its higher (10000), the mean of I_e diminishes from 0,024 to 0,022. However, this effect cannot be proven to be significant and is in this case just a coincidence. The opposite trend is shown for the effect of T_0, i.e. when it is changed from its lower value (100) to its higher (500) the mean of I_e increases from 0,0225 to 0,0235.

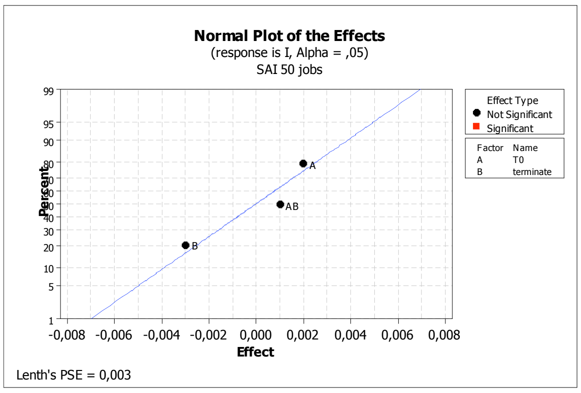

Figure 9.8. Normal probability plot of effects on I_e, *RSAI*, 50 jobs.

The Figure 9.8 shows that the individual effects as well as the combined effect of the two parameters are close to 0, and they are all considered being no significant given an alpha of 0.05.

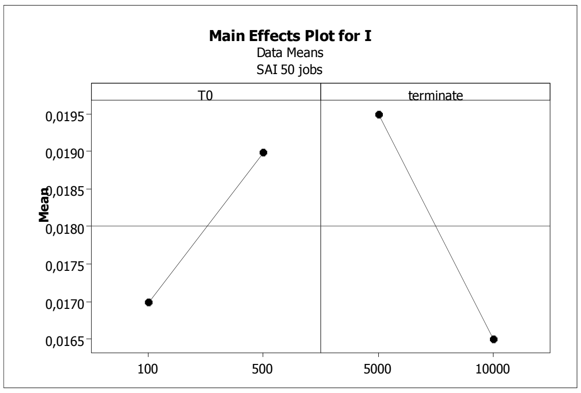

Figure 9.9. Plot of main effects on I_e, *RSAI*, 50 jobs.

As can be observed in the Figure 9.9, when for example *terminate* is changed from its lower value (5000) to its higher (10000), the mean of I_e diminishes from 0,0195 to 0,0165. However, this effect cannot be proven to be significant and is in this case just a coincidence. The opposite trend is shown for the effect of T_0, i.e. when it is changed from its lower value (100) to its higher (500) the mean of I_e increases from 0,017 to 0,019.

By examining the normal probability plots it is possible to discover that within this proposed range (T_0 between 100 and 500, *terminate* between 5000 and 10000) the parameters have no significant effect on I_e for none of the procedures or configurations.

Therefore a decision was made to establish the value of the parameters so that if favoured the running time. An observation was made regarding the parameter *terminate*; giving it a higher value, the running time increased significantly in all cases. The value of the parameter was therefore set to 5000.

Regarding the parameter T_0, it can be observed in the plot of the main effects that it seems that SAS performs better with the value set to 500 and SAI performs better with the value set to 100. Even though this effect could not be demonstrated to be significant, it was thought to be a better solution establishing the value of the parameters as mentioned rather than in any other way.

The parameters were set as demonstrated in Table 9.12 in the rest of the computational experimentation. No difference was made between the configurations of jobs or machines, nor the procedure used for generating the initial solution, according to the demonstrated results. When using SASI, the parameters were set according to each procedure used, first SAS and then SAI.

Table 9.12. The parameters established for the different procedures.

Procedure	T_0	***terminate***
SAS	500	5000
SAI	100	5000

9.5 Phase 2: Performance Experiments

9.5.1 Purpose

The objective of this phase, consisting of a number of performance experiments, was to deal with the hypotheses earlier stated:

- **H2**: The proposed procedure, the Simulated Annealing, needs a good initial solution to perform better while dealing with the stated problem considering the bicriteria.
- **H3**: Any of the proposed methods for generating neighbour solutions leads to better results than the others in terms of the value of the objective function of the proposed solution schedule.

To obtain statistically verifiable results, all five configurations of jobs and machines represented through the 600 instances, were processed five times each by the each of the six algorithms. The performance index I_e was then calculated for each instance comparing the best overall result obtained (among the 30) to the average result (mean value) for each algorithm:

$$I_e = \frac{\overline{F_\ell} - [\min]F_\ell}{[\min]F_\ell} \qquad \text{(Eq. 9.3)}$$

An average (mean) of I_e as well as its variance was then calculated for each algorithm and each of the five configurations, giving the possibility to compare the performance of the procedures in each case. Another parameter, the running time, was also considered when analyzing the performance.

9.5.2 Results

The performance of each procedure is shown in Figure 9.10-24. The index I_e is shown for each of the instances processed. Each graph consists of two series of data, the I_e for the procedure using EDD for creating the initial solution and the I_e for the procedure using Random. This way it is lucid to see weather one procedure performs better than the other in general. To be able to compare the different versions of the SA algorithm the figures have been organised so that the job and machine configurations are presented in order, beginning with the configuration of 15 jobs and 2 machines and finishing with the configuration of 50 jobs and 4 machines. For each configuration, three figures are presented, one for each version of the SA algorithm. For examples of the output data as well as the calculations, see Appendix D, Phase 2: Performance Experiments.

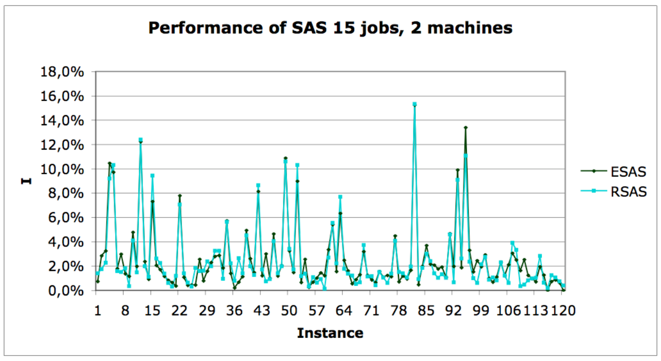

Figure 9.10. Overview of the performance of SAS 15 jobs, 2 machines.

As shown in the Figure 9.10, the procedures perform well for most instances, but have some peaks where they perform worse in comparison to the best result obtained in some few specific cases. However, the two procedures ESAS and RSAS follow each other regarding the performance in most cases.

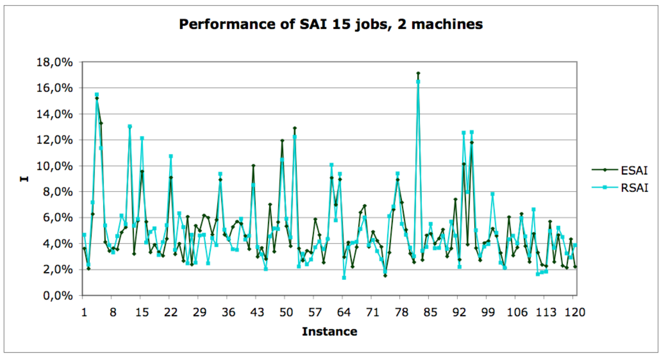

Figure 9.11. Overview of the performance of SAI 15 jobs, 2 machines.

As shown in the Figure 9.11, the performance of the procedures ESAI and RSAI fluctuate quite much, and have some peaks where they perform much worse in comparison to the best result obtained in some specific cases.

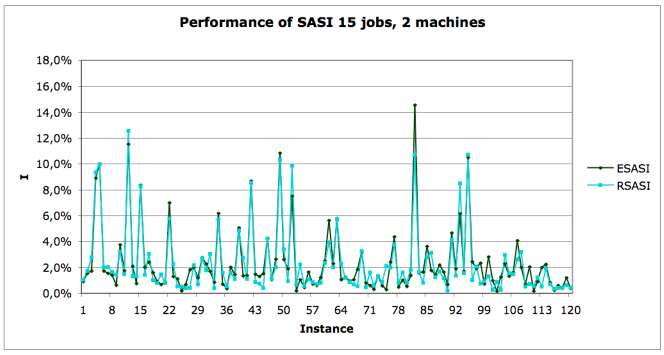

Figure 9.12. Overview of the performance of SASI 15 jobs, 2 machines.

As shown in the Figure 9.12, the procedures perform well in comparison to the best result obtained for most instances, but have some peaks where they perform worse in some few specific cases. However, the two procedures ESASI and RSASI follow each other regarding the performance in general.

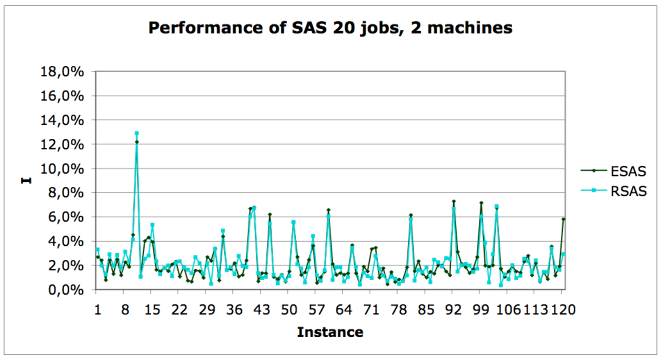

Figure 9.13. Overview of the performance of SAS 20 jobs, 2 machines.

The procedure shows several peaks, as shown in Figure 9.13, where the performance is worse than the best result obtained. The ESAS and RSAS show no significant difference in terms of performance.

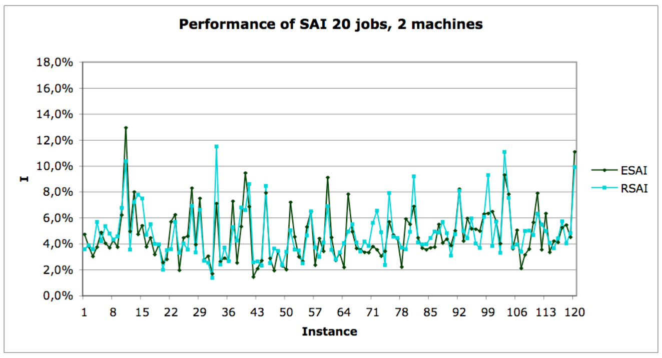

Figure 9.14. Overview of the performance of SAI 20 jobs, 2 machines.

The graphs shown in Figure 9.14 demonstrate a lot of peaks indicating a performance worse than the best result obtained. For some instances, RSAI seems to show lower values of I_e but in general the two procedures are showing a similar performance.

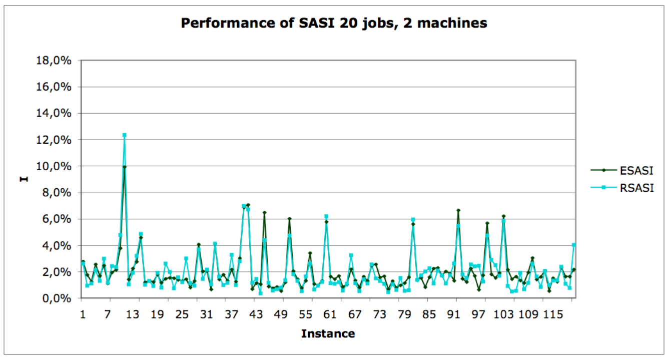

Figure 9.15. Overview of the performance of SASI 20 jobs, 2 machines.

As observed in Figure 9.15, ESASI and RSASI demonstrate low I_e for most instances.

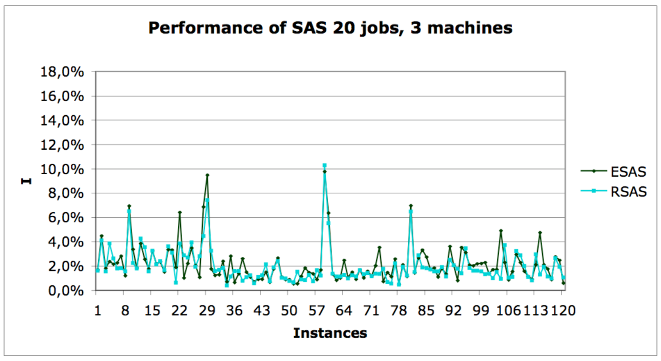

Figure 9.16. Overview of the performance of SAS 20 jobs, 3 machines.

The procedures show in Figure 9.16 that they perform well, with low I_e for most instances, but in a few cases they demonstrate a slightly worse performance.

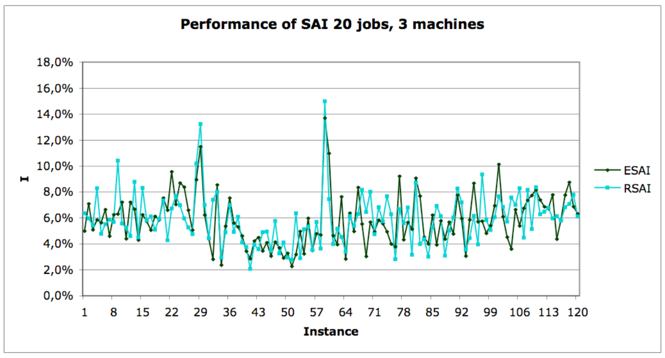

Figure 9.17. Overview of the performance of SAI 20 jobs, 3 machines.

In Figure 9.17 ESAI and RSAI show a fluctuating performance some percent worse than the best result obtained.

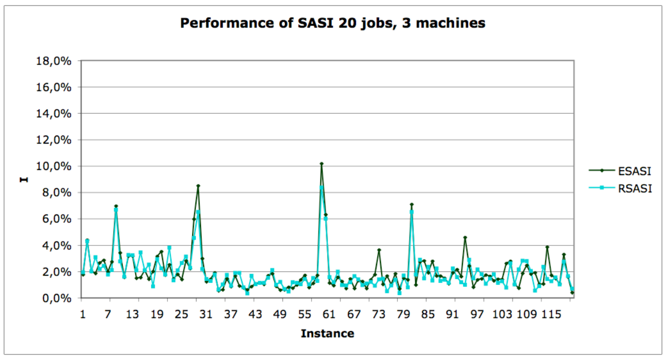

Figure 9.18. Overview of the performance of SASI 20 jobs, 3 machines.

The performance shown in Figure 9.18 by ESASI and RSASI is high in comparison to the best result obtained, with a just a few exceptions.

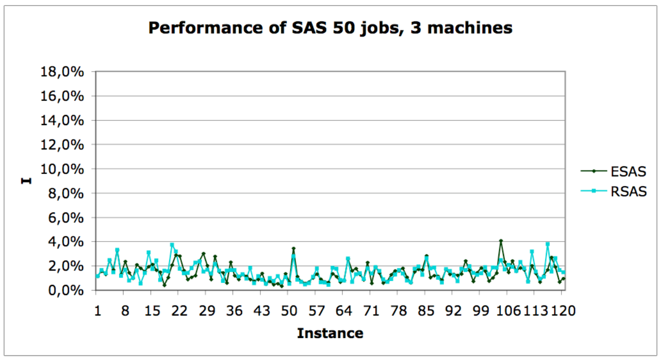

Figure 9.19. Overview of the performance of SAS 50 jobs, 3 machines.

The performance of both procedures is good in comparison with the best result obtained as can be observed in Figure 9.19 but it is not possible to distinguish ESAS from RSAS in terms of performance.

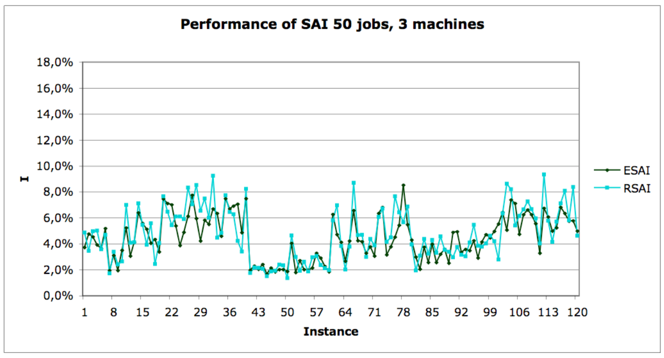

Figure 9.20. Overview of the performance of SAI 50 jobs, 3 machines.

The performance of ESAI and RSAI is fluctuating, but it is not possible to separate them in terms of performance as can be seen in Figure 9.20.

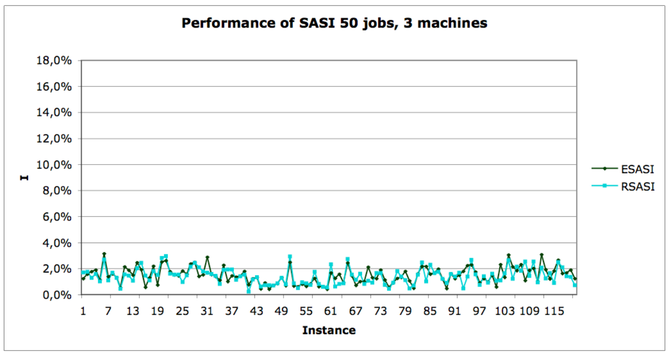

Figure 9.21. Overview of the performance of SASI 50 jobs, 3 machines.

The performance of both procedures is good in comparison with the best result obtained as can be observed in Figure 9.21 but it is not possible to distinguish ESAS from RSAS in terms of performance.

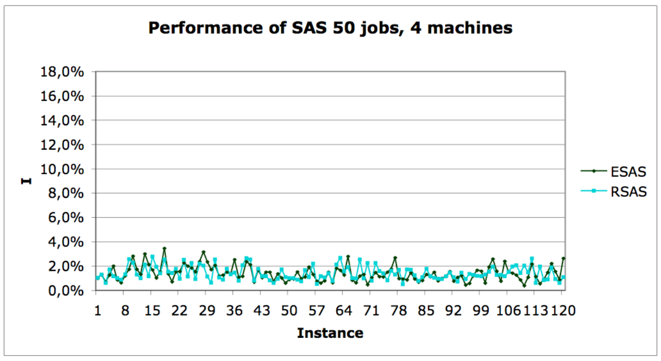

Figure 9.22. Overview of the performance of SAS 50 jobs, 4 machines.

The I_e of both procedures is low over all instances as can be observed in Figure 9.22 but it is not possible to distinguish ESAS from RSAS in terms of performance.

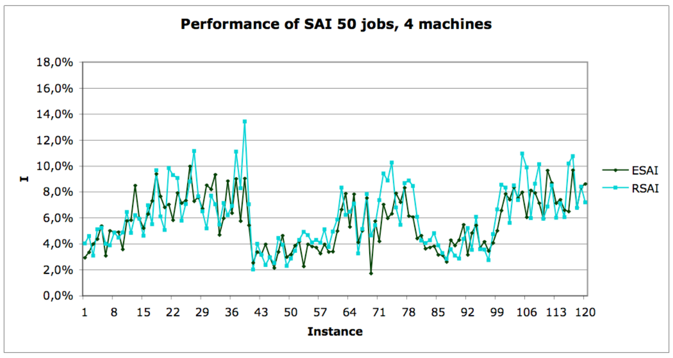

Figure 9.23. Overview of the performance of SAI 50 jobs, 4 machines.

The I_e fluctuates much over the instances and yet it is difficult to distinguish ESAI from RSAI in terms of performance as can be seen in Figure 9.23.

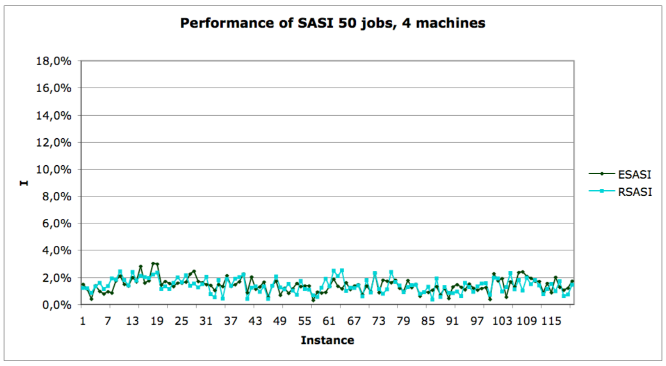

Figure 9.24. Overview of the performance of SASI 50 jobs, 4 machines.

Over the instances the I_e is low indicating a high performance for both ESASI and RSASI. However it is not possible to distinguish one method from the other in terms of performance.

A general observation is that it is not possible to distinguish any significant difference regarding the performance of the SA algorithm between the two methods used when creating an initial solution. In some isolated cases, one method is noticed to perform better than the other, but it is not always the same method that does so. In general, the two methods seem to show the same performance. Furthermore, SASI seems to perform best in general when observing and comparing the three different versions of the algorithm for each configuration.

Next, a summary of the results is presented in Table 9.13 and 9.14. For each procedure and configuration the average (mean) I_e and its variance are shown in Table 9.13 (the best results are marked in bold) and the average running time is presented in Table 9.14.

Table 9.13. Summary of the results for the 6 procedures, I_e (%) and its variance.

		ESAS		RSAS		ESAI		RSAI		ESASI		RSASI	
Jobs	**Machines**	I_e	σ^2	I_e	σ^2	I_e	σ^2	I_e	σ^2	I_e	σ^2	I_e	σ^2
15	**2**	2.60	0,0008	2.51	0,0008	5.07	0,0008	5.13	0,0008	2.39	0,0007	**2.30**	0,0007
20	**2**	2.27	0,0003	2.21	0,0003	4.68	0,0004	4.80	0,0004	2.06	0,0003	**2.01**	0,0003
20	**3**	2.21	0,0003	2.01	0,0002	5.82	0,0004	5.90	0,0004	2.00	0,0002	**1.91**	0,0002
50	**3**	1.46	0,00005	1.51	0,00005	4.49	0,0003	4.74	0,0004	1.49	0,00004	**1.42**	0,00004
50	**4**	1.40	0,00004	1.41	0,00003	5.70	0,0004	6.05	0,0005	1.42	0,00003	**1.39**	0,00003

It is possible to observe that SASI (both ESASI and RSASI) has slightly lower I_e and variance than SAS. Between EDD and Random it is again not possible to distinguish any significant difference for any of the three procedures (SAS, SAI and SASI). Generally SAI presents higher values than the other two procedures. For SAS and SASI there is a decreasing trend among I_e when the number of jobs and machines increases.

Table 9.14. Summary of the running times for the 6 procedures.

Jobs	**Machines**	ESAS	RSAS	ESAI	RSAI	ESASI	RSASI
15	2	5.1	5.0	6.2	6.0	11.4	10.9
20	2	6.6	6.5	8.2	8.3	14.5	14.7
20	3	9.2	9.0	11.7	12.0	21.4	21.0
50	3	20.1	20.6	27.0	26.6	46.5	47.3
50	4	29.1	28.6	38.2	37.9	67.4	66.7

It is possible to observe that the more complex problem, the greater is the running time. SAS also seems to be fastest followed by SAI and last SASI. Between the two procedures to create neighbours, EDD and Random, there seems to be no significant difference regarding the running time (it is possible to do this comparison observing for example ESAS and RSAS for each configuration).

Finally, commenting on the two stated hypotheses it seems that H2, regarding the initial solution, is false and that H3, regarding the generators of neighbour solutions, is true being SASI the best version of the heuristic.

9.6 Phase 3: Bicriteria Experiments

9.6.1 Purpose

The objective of this final phase was to deal with the hypothesis earlier stated:

- **H2**: The proposed procedure, the Simulated Annealing, needs a good initial solution to perform better while dealing with the stated problem considering the bicriteria.

This time the analysis was more focused on the performance of the SA algorithm when altering the bicriteria expressed through the linear convex objective function. The reason for this analysis is based on the situation where the production somehow needs to be altered due to external influence and the focus is more on one of the criteria in the objective function. Since a bicriteria objective function is applied, this adaptation is easy to implement. By modifying the α in the objective function it is possible to give more importance to one of the criteria than the other.

The experiments were realized using the procedure that had been performing best in the earlier experiments, SASI. Both methods when creating an initial solution were used and the experiments were realized with a smaller subgroup of 60 instances including the configurations of 50 jobs and 3 and 4 machines (30 instances each). These configurations represented the most complex problem and were therefore thought to be most relevant to use when realizing these experiments.

In one case the α in the objective function was set to 0.6 and in the other to 0.8 (β set to 0.4 and 0.2 respectively). To obtain statistically verifiable results, the two configurations of jobs and machines represented through the 60 instances, were processed five times each by the two algorithms. The performance index I_e was then calculated for each instance comparing the best overall result obtained (among the 10) to the average result for each algorithm:

$$I_e = \frac{\overline{F_\ell} - [\min]F_\ell}{[\min]F_\ell} \qquad \text{(Eq. 9.4)}$$

An average of I_e as well as its variance was then calculated for each algorithm and each of the two configurations, giving the possibility to compare the performance of the procedures in each case. Also another parameter, the running time, was considered when analyzing the performance.

9.6.2 Results

The performance of each procedure is shown in Figure 9.25-28. The index I_e is shown for each of the instances. For examples of the output data and the calculations, see Appendix D, Phase 3: Bicriteria Experiments.

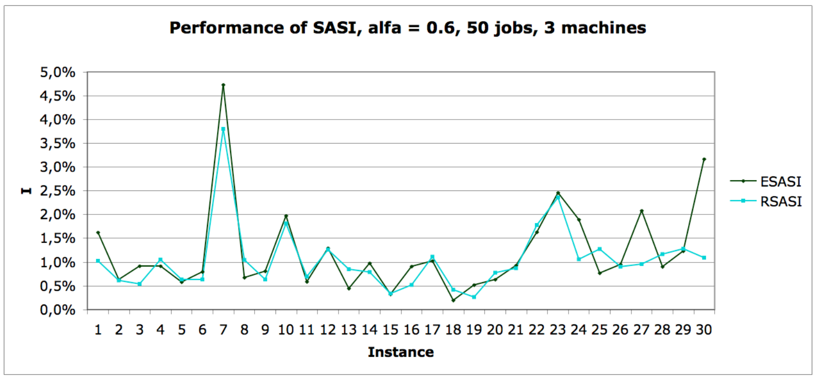

Figure 9.25. Overview of the performance of SASI, α = 0.6, 50 jobs, 3 machines.

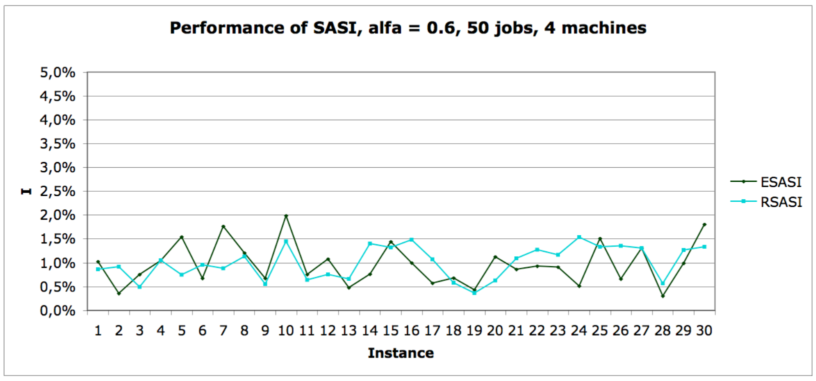

Figure 9.26. Overview of the performance of SASI, α = 0.6, 50 jobs, 4 machines.

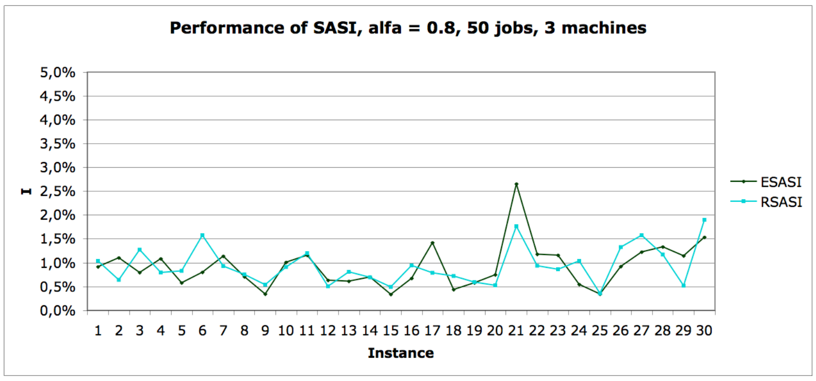

Figure 9.27. Overview of the performance of SASI, α = 0.8, 50 jobs, 3 machines.

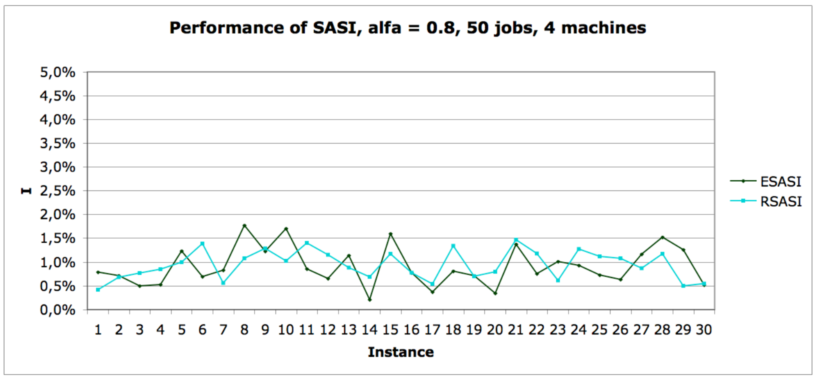

Figure 9.28. Overview of the performance of SASI, α = 0.8, 50 jobs, 4 machines.

In all four figures it is possible to observe a low overall I_e for both SASI procedures. It is just in one configuration when α = 0.6 that the performance is shown to decrease for one isolated case. Furthermore, it seems that no difference can be observed in terms of performance between the two methods for generating an initial solution; EDD and Random.

The results are also presented as a summary for each α in the Tables 9.15, 9.16 and 9.17. The average I_e (mean) and its variance are presented in the first two tables. In the third, the average running time for each configuration and procedure is presented.

Table 9.15. Results for α = 0.6.

		ESASI		**RSASI**	
Jobs	**Machines**	I_e	σ^2	I_e	σ^2
50	3	1.22 %	0,00009	1.05 %	0,00005
50	4	0.97 %	0,00002	1.01 %	0,00001

Table 9.16. Results for α = 0.8.

		ESASI		**RSASI**	
Jobs	**Machines**	I_e	σ^2	I_e	σ^2
50	3	0.93 %	0,00002	0.93 %	0,00002
50	4	0.91 %	0,00002	0.94 %	0,00001

The procedures present low I_e and low variance as well for both α. No significant difference can be observed between the two methods for generating initial solutions in either case.

Table 9.17. The running time for the bicriteria experiments.

		α = 0.6		**α = 0.8**	
Jobs	**Machines**	**ESASI**	**RSASI**	**ESASI**	**RSASI**
50	3	41.6	42.4	38.0	38.2
50	4	60.4	60.8	55.3	52.8

It is possible to observe a slight difference between α = 0.6 and α = 0.8, where the procedures seem to have a greater running time for the earlier case.

Commenting on the stated hypothesis, H2, it seems to be false after having realized these experiments as well.

10 EXTERNAL IMPACT OF PROJECT

10.1 Budget

No assumptions will be made of the probable effect on cost reduction the study could have, if some of its parts were implemented in a real world industry. However, the cost of the study itself has been looked at, and the following entries are accounted for:

- Direct labour
- Material cost of report
- Unforeseen expenses

The direct labour accounted for consists of subject research, development of heuristic and realization of experiments. The work has been performed using a personal computer and several computers belonging to the university. As a student, the cost of this usage is considered absent. As for the direct labour, an estimation has been done to account for all the hours spent on the realization. It has been estimated that during its approximately 27 weeks, 4 hours per day (including weekends) was needed for the completion of the study. If the cost per hour is compared to that of a junior consultant, i.e. a recent graduated engineer, 45€ per hour will result in the total cost of direct labour demonstrated in Table 10.1. Furthermore, the material cost of the report is shown in Table 10.2. The report was printed in 5 copies.

Table 10.1. Estimation of the cost of direct labour during the duration of the project.

Weeks	*27*
Hours per day (h)	*4*
Cost per hour (€ / h)	*45*
Total cost	***34 020 €***

Table 10.2. Estimation of the material cost of the report. Cost demonstrated per copy.

Pages	*140*
Printing cost (€/page)	*0.05*
Binding (€/booklet)	*5*
CD (€/disk)	*0.9*
Total	***12.9€***

The unforeseen expenses were estimated to approximately 100€. The total cost of the study is given in Table 10.3.

Table 10.3. Estimation of the material cost of the report.

Direct labour	*34 020*
Report material (5 copies)	*64.5*
Unforeseen expenses	*100*
Total	***34 184.5€***
VAT (16 %)	*5 469.5 €*
Total expenses	***39 654 €***

10.2 Environmental Impact

Without doing any deeper investigations about the effect the study could have on an industry and its environmental impact if implemented there, one could easily assert that the study as such has not caused any more effect on the environment than an ordinary university semester for an average student. The only impact accounted for was due to:

- Printing paper of the report (printed on ecofriendly paper)
- The ink used in the printing
- The CDs used for the storage of the report and extras.

Furthermore, it is worth mentioning that if implemented in a manufacturing facility, the heuristic developed could effect the industry's operations and their effect upon the environment in the following manner:

- Energy consumption: making the manufacturing process more efficient also causes the reduction of energy usage for that process.
- Diminishing the setup times can result in a more efficient way of configuring the machines and hence a reduction in the use of expandable material.
- A production with a high level of service can establish a more efficient way of distributing its products, which could result in a decrease in transports and the fuel consumption and contamination that it brings.

11 CONCLUSIONS

Through the different phases of the experimentation, from establishing the procedure parameters to realizing the bicriteria experimentation, it has been possible to develop a well working algorithm that can find a good solution within a feasible amount of time. During this process, a set of algorithms has been tested to see which one performs best and which one is the most appropriate when using the bicriteria objective function.

In the pre-phases it was possible to prove that the EDD priority rule outperformed the other procedures when creating an initial solution. However, when observing the figures presenting the performance of the procedures, comparing each one when using EDD and when using Random, it is seems that the Simulated Annealing procedure proposed does not need a good initial solution to be able to perform well while dealing with the stated scheduling problem and the provided collection of instances. The same conclusion can be drawn when observing the result summary presented in the tables. In the figures, one can observe that for individual instances one procedure some times performs slightly better than the other, but in general no distinguishing can be made between the two regarding their influence on the overall performance.

The results from the main experiments also allow the comparison between the three versions used when generating neighbour solutions. The one that performs slightly better is SASI, using both the other methods together, but on the other hand it spends more time on finding the best solution.

When comparing the SAS and SAI procedures, SAS is the better one. It performs better in all configurations, working faster to find the best solution. It seems that this method of generating neighbour solutions (Swap) is preferable when using the proposed SA algorithm for this specific problem.

In the last phase, once again the hypothesis about weather or not the SA algorithm needed a good initial solution to perform well was tested. But this time the SASI procedure was tested when altering the importance of the criteria in the objective function. Since it had been shown that this procedure performed better than the others in earlier realized experiments, it was tested in two additional situations when α was set to 0.6 and 0.8. The results are the same as in the situation with α set to 0.4, i.e. the conclusion is that the proposed SA algorithm does not need a good initial solution to be able to find a good solution.

11.1 Suggestions and Future Work

During the realization of the study, some ideas also emerged that could not be further investigated due to the limitations of the study and therefore are presented here as ideas to develop if given time and having the interest in the future.

With the procedures proposed for creating an initial solution, it seems that the SA algorithm does perform neither better nor worse. These procedures, however, are based on priority rules and are static and fairly simple in their nature, and therefore it could be interesting to test if the SA algorithm would demonstrate the same behaviour if a more advanced procedure was implemented to create the initial solution. One suggestion would be to implement a dynamic priority rule to be able to create an initial solution that perhaps could favour the bicriteria in a better way.

Furthermore, the performance of the SA algorithm seems to depend on the method used when generating the neighbours. One suggestion for future work could be to test the proposed SA algorithm with other methods for generating neighbour solutions to see if the performance would improve. Using various methods at the same time, generating simultaneously a neighbour solution each could be one approach, letting the SA algorithm decide which one to use and weather or not to accept it. Other options could be to use one method to generate the neighbours and to switch to others when considered appropriate, having various criteria of when to switch. Developing another decreasing schedule for the temperature could also be an alternative.

The nature of the heuristic allows all these suggestions to be implemented without any mayor difficulties.

ACKNOWLEDGEMENTS

The author wishes to thank his mentor, Imma Ribas, for her distinct guidance through this project, and especially for her obvious support in those moments when the author lacked both time and motivation.

The gratitude of the author is also directed to his football team, L@s Piltrafillas, for the joy the members of the team spread in times of victory but also in moments not so victorious.

REFERENCES

[1] *A Branch and Bound Algorithm for a Single-machine Scheduling Problem with Positive and Negative Time-lags*
P. Brucker, T. Hilbig and J. Hurink
Discrete Applied Mathematics **94**, 77-99 (1999)

[2] *A Branch-and-bound Algorithm for the Exact Solution of the 3-Machine Scheduling Problem*
Z.A. Lomnicki
Operational Research Quarterly **16**, 89-100 (1965)

[3] *A Concise Survey of Scheduling with Time-dependent Processing Times*
T.C.E. Cheng, Q. Ding and B.M.T. Lin
European Journal of Operational Research **152**, 1-13 (2004)

[4] *A Decomposition Algorithm for the Single Machine Total Tardiness Problem*
C. N. Potts and L. N. Van Wassenhove
Operations Research Letters **1**, 177-181 (1982)

[5] *A Multi-Exchange Neighborhood for Minimum Makespan Parallel Machine Scheduling Problems*
A. Frangioni, E. Necciari and M.G. Scutellà
Journal of Combinatorial Optimization **8**, 195-220 (2004)

[6] *An Exact Algorithm for the Identical Parallel Machine Scheduling Problem*
E. Mokotoff
European Journal of Operational Research **152**, 758-769 (2004)

[7] *A Scheduling Problem with Unrelated Parallel Machines and Sequence Dependent Setups*
M.G. Ravetti, G.R. Mateus, P.L. Rocha and P.M. Pardalos
International Journal of Operational Research **2**, 380-399 (2007)

[8] *A Simulated Annealing Approach to Makespan Minimization on Identical Parallel Machines*
W.-C. Lee, C.-C. Wu and P. Chen
The International Journal of Advanced Manufacturing Technology **31**, 328-334 (2006)

[9] *A State-of-the-art Review of Parallel-machine Scheduling Research*
T.C.E. Cheng and C.C.S. Sin
European Journal of Operational Research **47**, 271-292 (1990)

[10] *A Survey of Scheduling Problems with Setup Times or Costs*
A. Allahverdi, C.T. Ng, T.C.E. Cheng and M.Y. Kovalyov
European Journal of Operational Research **187**, 985-1032 (2008)

[11] *Computer-aided Design, Engineering and Manufacturing; System Techniques and Applications Volume 1: System Techniques and Computational Methods*
C.T. Leondes
CRC Press, (2002)

[12] *Equation of State Calculations by Fast Computing Machines*
N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller
Journal of Chemical Physics **21**, 1087-1092 (1953)

[13] *Generating Efficient Schedule for Identical Parallel Machines Involving Flow-time and Tardy Jobs*
J.N.D. Gupta and A.J. Ruiz-Torres
European Journal of Operational Research **167**, 679-695 (2005)

[14] *Handbook of Metaheuristics*
Edited by F. Glover and G.A. Kochenberger
Kluwer Academic Publishers (2003)

[15] *Introduction to Operations Research*
F.S. Hillier and G.J. Lieberman
McGraw-Hill (2006)

[16] *Local Search in Combinatorial Optimization*
E.H.L. Aarts and J.K. Lenstra
Princeton University Press (2003)

[17] *Monte Carlo Strategies in Scientific Computing*
J.S. Liu
Springer, New York (2001)

[18] *Multicriteria Scheduling*
H. Hoogeveen
European Journal of Operational Research **167**, 592-623 (2005)

[19] *Multicriteria Scheduling: Theory, Models and Algorithms*
V. T'Kindt and J.C. Billaut
Springer (2002)

[20] *Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey.*
R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan
Annals of Discrete Mathematics **5**, 287-326 (1979)

[21] *Parallel Simulated Annealing and Evolutionary Selection for Combinatorial Optimization*
V. Delport
Electronics Letters **34**, 758-759 (1998)

[22] *Planning and Scheduling in Manufacturing and Services*
M.L. Pinedo
Springer, New York (2005)

[23] *Scheduling Computer and Manufacturing Processes*
J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt and J. Weglarz
Springer, Berlin (2001)

[24] *Scheduling Jobs on Parallel Machines with Sequence-dependent Setup Times*
Y.H. Lee and M. Pinedo
European Journal of Operational Research **100**, 464-474 (1997)

[25] Secuenciación Vol. 1
R. Companys Pascual
CPDA, ETSEIB, Barcelona (2003)

[26] *Simulated annealing heuristics for the average flow-time and the number of tardy jobs bi-criteria identical parallel machine problem*
A.J. Ruiz-Torres, E.E. Enscore and R.R. Barton
Computers & Industrial Engineering **33**, 257-260 (1997)

[27] *Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing*
D.-W. Kim, K.-H. Kim, W. Jang and F.F. Chen
Robotics and Computer Integrated Manufacturing **18**, 223-231 (2002)

Additional Utilities

ISI Web of Science®
http://www.accesowok.fecyt.es/isip/

APPENDIX

Appendix A

Example of a set of instances *D* with 15 jobs and 2 machines (LOI32588).

15	2	100	.5	.8	.8	10								
10	3	4	8	7	2	6	4	8	6	4	2	1	4	18
0	21	18	19	10	16	6	8	24	24	14	14	17	4	24
31	0	15	12	26	26	21	21	22	24	23	22	14	21	27
25	3	0	9	11	16	6	13	8	14	8	17	9	16	25
22	17	20	0	31	14	25	9	22	26	16	12	18	25	22
16	22	19	20	0	16	22	20	24	14	15	20	12	6	21
23	12	18	21	25	0	11	4	8	14	14	10	13	20	24
29	30	28	20	19	28	0	23	25	31	24	20	12	19	26
19	17	14	17	25	8	19	0	16	22	10	6	9	16	20
23	14	11	18	18	17	17	19	0	6	14	22	14	20	31
25	15	12	12	12	16	18	13	20	0	8	16	8	15	31
23	7	4	13	15	8	10	5	12	18	0	11	13	20	25
27	11	8	17	19	12	14	9	16	22	4	0	17	24	24
17	19	16	8	27	19	22	17	13	19	12	8	0	7	23
10	17	14	15	20	12	16	15	20	21	10	21	15	0	20
25	4	10	16	16	2	13	6	10	16	6	2	15	21	0
10														
1														
32	41	45	85	63	8	49	40	97	60	58	56	34	61	82
0	0	0	0	38	81	81	93	113	131	132	138	156	289	291
2														
15	33	35	63	86	67	84	48	77	8	20	69	66	5	21
0	0	0	0	0	18	55	92	175	198	242	244	246	290	308
3														
22	88	99	68	23	93	5	75	72	17	24	13	57	93	83
0	0	61	64	104	137	146	149	164	203	213	222	257	293	295
4														
6	22	32	36	92	33	20	85	16	90	58	17	6	11	88
0	0	0	0	0	2	68	94	95	128	130	259	297	302	317
5														
39	73	98	6	41	61	91	73	27	73	78	21	12	28	93
0	0	2	4	9	39	47	51	87	153	158	186	197	199	264
6														
72	74	81	65	14	71	72	38	67	86	39	27	68	5	15
0	0	0	76	82	85	124	169	197	205	212	256	275	327	330
7														
33	35	41	51	77	91	93	88	85	84	26	76	51	34	34
0	0	0	0	0	83	127	192	220	228	236	250	279	296	317
8														
21	32	44	48	78	62	97	29	93	32	29	46	49	65	14
0	0	0	0	40	139	193	203	228	233	256	261	281	291	303
9														
23	26	31	49	54	54	79	100	24	39	64	87	92	22	67
0	0	0	0	0	0	0	0	42	49	53	79	105	158	276
10														
10	45	15	52	78	35	79	95	18	32	43	92	52	77	100
0	0	108	118	118	152	154	234	262	268	275	277	280	285	322

First line:

n	m	max p_j	λ	α	β	n° of total specimen

S_{jk}
Matrix with setup times where the first line indicates the setup time for the first job scheduled when the machine is in stand by. The second line indicates the setup time between job n° 1 and each of the rest, etc.

n° of total instances in the set of instances

n° of the specimen

Processing time for each job in this specimen, p_j

Due date for each job in this specimen, d_j

Symbol	Explanation
n	n° of jobs
m	n° of machines
max p_j	The maximum processing time possible
λ	Control parameter for the setup times
α	Control parameter for the due dates
β	Control parameter for the due dates

Appendix B

Example of an instance processed presented by its initial solution, best solution and other captured data. For explanations to abbreviations, see Chapter 1 Abbreviations and Symbols.

Example of 50 jobs, 4 machines processed with ESASI.

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	Cmaxbest	Tmaxbest	Time (s)	it
50	4	1	100	0,8	0,8	0,8	10	475	428,4	433,8	756	219	72,5	517652

Initial Solution

Cmax	Tmax	finit
832	237	475

Solution Sequence

M1	M2	M3	M4
1	2	3	4
5	6	7	8
9	11	12	13
10	15	16	17
14	20	18	19
23	26	21	22
27	30	24	25
31	34	29	28
35	38	32	33
39	43	36	37
40	45	41	42
44		47	48
46		49	
		50	

Best Solution

M1	M2	M3	M4
14	5	17	8
3	9	12	10
1	21	4	11
6	7	2	18
23	13	16	24
25	15	20	26
28	19	22	31
32	27	34	42
38	30	29	35
37	36	33	40
39	41	43	46
49	50	45	
47		44	
		48	

Example of 50 jobs, 4 machines processed with RSASI.

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	Cmaxbest	Tmaxbest	Time (s)	it
50	4	1	100	0,8	0,8	0,8	10	822,6	425,6	431	740	225	86,6	617466

Initial Solution

Cmax	Tmax	finit
843	809	822,6

Solution Sequence

M1	M2	M3	M4
19	30	28	10
42	18	16	21
26	5	11	29
47	46	8	6
23	27	20	9
4	17	35	36
1	22	41	13
34	7	43	40
12	44	33	25
45	31	39	3
15	32	48	38
14	24	2	50
37	49		

Best Solution

M1	M2	M3	M4
12	9	4	3
11	21	2	14
6	13	7	5
22	8	10	1
18	30	15	16
23	19	17	20
28	25	24	27
35	26	29	34
40	32	31	47
49	33	43	36
38	37	41	42
39	50	48	46
45	44		

Appendix C

The code for the metaheuristic as it was written in Visual Basic© 6.0. The metaheuristic is available on the CD for tests.

```
Option Explicit
Dim n As Single                'number of jobs
Dim m As Single                'number of machines
Dim s() As Integer             'set-up times
Dim ne As Single               'number of examples
Dim ex As Single               'the number of the current example
Dim j() As Integer             'job number
Dim jdata() As Integer
Dim p() As Integer             'processing time for each job
Dim pdata() As Integer
Dim d() As Integer             'due date for each job
Dim ddata() As Integer
Dim c() As Integer             'completion time for each job
Dim L() As Integer             'lateness for each job
Dim T() As Integer             'tardiness for each job
Dim tempC() As Integer
Dim Counter() As Single
Dim Cseq() As Integer
Dim solinit() As Integer       'initial solution
Dim solcurr() As Integer       'current solution
Dim solneigh() As Integer      'neighbour solution
Dim soltemp() As Integer       'temporary neighbour solution under certain
conditions in the insert method
Dim solbest() As Integer       'best sequence
Dim finit As Single            'initial objective function
Dim fcurr As Single            'result for best obtained solution
Dim fneigh As Single           'weighted objective function
Dim fbest As Single            'best solution (function objective)
Dim fbesthalf As Single
Dim SumC As Single             'total completion time for all jobs in all
machines
Dim SumT As Single             'total tardiness for all jobs in all
machines
Dim Cmax As Single
Dim Cmaxneigh As Single
Dim Cmaxbest As Single
Dim Tmax As Single
Dim Tmaxneigh As Single
Dim Tmaxbest As Single
Dim alfa As Single             'porcentage determining the weighted
criteria
Dim beta As Single             'decrease in Tr for every iteration
Dim delta As Single            'fneigh-fcurr
Dim Tr As Double               'starting temperature for SA algorithm
Dim Tinit As Double
Dim it As Single               'iterations
Dim iter As Single             'stop criteria
Dim change As Single           'indicates at what point the insert
algorithm should be called upon
Dim terminate As Single        'calculator
Dim equal As Single
Dim same As Single             'counts times that delta=0
Dim fixinsert As Single
Dim filepath As String
Dim collection As Integer
Dim fileID As Integer
Dim filename As String
Dim Time As Single              'variables to measure time
Dim startTime As Single
Dim endTime As Single
Dim min As Single
Dim Index As Single
Dim mpos As Single
Dim ppos As Single
Dim upprepningar As Single
Dim initmethod As Single
Dim neighbourmethod As Single
Dim p1 As Single, p2 As Single, p3 As Single, p4 As Single, p5 As Single
Dim i As Single, k As Single, x As Single, y As Single, z As Single, v As
Single   'help variables

Private Sub Dir1_Change()
  File1.Path = Dir1.Path
End Sub
Private Sub Drive1_Change()
   Dir1.Path = Drive1.Drive
End Sub
Private Sub File1_Click()
proceduremenu.Enabled = True
instance.Enabled = True
End Sub
Private Sub number1_Click()
ex = 1
number1.Checked = True
number2.Checked = False
number3.Checked = False
number4.Checked = False
number5.Checked = False
number6.Checked = False
number7.Checked = False
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number10_Click()
ex = 10
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = False
number5.Checked = False
number6.Checked = False
number7.Checked = False
number8.Checked = False
number9.Checked = False
number10.Checked = True
End Sub
Private Sub number2_Click()
ex = 2
number1.Checked = False
number2.Checked = True
number3.Checked = False
number4.Checked = False
number5.Checked = False
number6.Checked = False
number7.Checked = False
```

```
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number3_Click()
ex = 3
number1.Checked = False
number2.Checked = False
number3.Checked = True
number4.Checked = False
number5.Checked = False
number6.Checked = False
number7.Checked = False
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number4_Click()
ex = 4
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = True
number5.Checked = False
number6.Checked = False
number7.Checked = False
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number5_Click()
ex = 5
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = False
number5.Checked = True
number6.Checked = False
number7.Checked = False
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number6_Click()
ex = 6
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = False
number5.Checked = False
number6.Checked = True
number7.Checked = False
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number7_Click()
ex = 7
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = False
```

```
number5.Checked = False
number6.Checked = False
number7.Checked = True
number8.Checked = False
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number8_Click()
ex = 8
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = False
number5.Checked = False
number6.Checked = False
number7.Checked = False
number8.Checked = True
number9.Checked = False
number10.Checked = False
End Sub
Private Sub number9_Click()
ex = 9
number1.Checked = False
number2.Checked = False
number3.Checked = False
number4.Checked = False
number5.Checked = False
number6.Checked = False
number7.Checked = False
number8.Checked = False
number9.Checked = True
number10.Checked = False
End Sub
Private Sub Option1_Click()
alfa = 0.4
End Sub
Private Sub Option2_Click()
alfa = 0.6
End Sub
Private Sub Option3_Click()
alfa = 0.8
End Sub

Private Sub random_Click()
initmethod = 4
EDD.Checked = False
SPT.Checked = False
mixed.Checked = False
random.Checked = True
If SAS.Checked = True Or SAI.Checked = True Or SASI.Checked = True Then
solve.Enabled = True
End If
End Sub

Private Sub SAI_Click()
neighbourmethod = 2
SAS.Checked = False
SAI.Checked = True
SASI.Checked = False
If EDD.Checked = True Or SPT.Checked = True Or mixed.Checked = True Or
random.Checked = True Then
solve.Enabled = True
```

```
End If
End Sub

Private Sub SAS_Click()
neighbourmethod = 1
SAS.Checked = True
SAI.Checked = False
SASI.Checked = False
If EDD.Checked = True Or SPT.Checked = True Or mixed.Checked = True Or
random.Checked = True Then
solve.Enabled = True
End If
End Sub

Private Sub SASI_Click()
neighbourmethod = 3
SAS.Checked = False
SAI.Checked = False
SASI.Checked = True
If EDD.Checked = True Or SPT.Checked = True Or mixed.Checked = True Or
random.Checked = True Then
solve.Enabled = True
End If
End Sub

Private Sub start_click()
display1.Cls
display1.Print "Follow these steps to schedule a given problem:"
display1.Print "1. Select an archive with a setup of instances."
display1.Print "2. Select a method to generate an initial solution"
display1.Print "from the procedure menu."
display1.Print "3. Select a method to generate the neighbour solutions"
display1.Print "from the procedure menu."
display1.Print "4. Select an instance (the first is selected by default)"
display1.Print "from the instance menu."
display1.Print "5. Select the value of alfa (0.4 is selected by default)."
display1.Print "6. Click Solve."
display1.Print ""
display1.Print "If you want to repeat the same procedure again,"
display1.Print "click Solve. If you want to change some configuration,"
display1.Print "do so via the filelist or the menus and click Solve."
display1.Print "All info of the last instance scheduled"
display1.Print "is also stored in the file Results.dat"
File1.Enabled = True
End Sub

Private Sub EDD_Click()
initmethod = 1
EDD.Checked = True
SPT.Checked = False
mixed.Checked = False
random.Checked = False
If SAS.Checked = True Or SAI.Checked = True Or SASI.Checked = True Then
solve.Enabled = True
End If
End Sub

Private Sub End_Click()
End
End Sub
'read data from file and execute algorithms
Private Sub solve_click()
```

```
Open App.Path & "\Results.dat" For Output As #1
filename = File1.Path & "\" & File1.filename
fileID = FreeFile
Open filename For Input As #fileID
beta = 0.999                          'decrease factor for the temperature
schedule
If Option1.Value = True Then
alfa = 0.4
End If
Dim a(1 To 7) As Single
For i = 1 To 7
   Input #fileID, a(i)
Next i
n = a(1)                                        'stores number of jobs
m = a(2)                                        'stores number of machines
If n > 50 Then
display1.Print "The number of jobs are too great!"
GoTo error:
End If
If m > 4 Then
display1.Print "The number of machines are too great!"
GoTo error:
End If
p1 = a(3): p2 = a(4): p3 = a(5): p4 = a(6): p5 = a(7)
ReDim s(n + 1, n)
ReDim j(n): ReDim jdata(n)
ReDim p(n): ReDim pdata(n)
ReDim d(n): ReDim ddata(n)
For i = 0 To n                                  'read set-up times
   For k = 1 To n
   Input #fileID, s(i, k)
   Next k
Next i
Input #fileID, ne                                    'number of examples
v = 0
If number1.Checked = True Then
ex = 1
End If
Do While v < ex
Input #fileID, v                                'number of current example
For i = 1 To n
   j(i) = i
   jdata(i) = j(i)
Next i
For i = 1 To n
   Input #fileID, p(i)                 'stores processing time for the jobs
   pdata(i) = p(i)
Next i
For i = 1 To n
   Input #fileID, d(i)                       'stores due dates for the jobs
   ddata(i) = d(i)
Next i
Loop
v = 0
If initmethod = 1 Then
Call orderEDD
Call asign
End If
If initmethod = 2 Then
Call orderSPT
Call asign
End If
```

```
If initmethod = 3 Then
Call ordermixed
Call asign
End If
If initmethod = 4 Then
Call asignrdm
End If
startTime = Timer
Randomize
Call SA
error:
Close #fileID                                        'closing data file
Close #1
End Sub

'chooses the sequence via SA algorithm
Private Sub SA()
d = ddata: p = pdata                   'restore initial values (not ordered)
Dim prob As Single

If neighbourmethod = 1 Or neighbourmethod = 3 Then
Tinit = 500
End If
If neighbourmethod = 2 Then
Tinit = 100
End If

it = 0
same = 0
Tr = Tinit
change = 0
iter = 0
fixinsert = 0
terminate = 5001

Do While iter < terminate And it < 1000000          'termination criteria
If Tr < 0.12 Then
Tr = 0.2
End If
it = it + 1
If neighbourmethod = 1 Then
Call swap
End If
If neighbourmethod = 2 Then
Call insert
End If
If neighbourmethod = 3 Then
If change < 5000 Then
Call swap
Else
Call insert
End If
If change = 5000 Then
Tr = 100: fbesthalf = fbest: fbest = 10000: fixinsert = 1: same = 0: iter =
0
End If
End If

delta = fneigh - fcurr                          'calculates delta
If delta > 0 Then                               'SA element
prob = Exp(-delta / Tr)
Else
```

```
prob = 0
End If
If delta = 0 And same < 1000 Then
equal = Rnd(): same = same + 1
Else
equal = 0
End If

'SA algorithm decides wheater to pic the new solution or not
If delta < 0 Or prob > Rnd() Or equal > Rnd() Then
solcurr = solneigh: fcurr = fneigh: Tr = Tr * beta
'Temperature is decreasing for every solution proposed
End If
If delta < 0 Then
change = 0: iter = 0
Else
change = change + 1: iter = iter + 1
End If
If fixinsert > 0 Then                              'after change has been
met, insert is called inst
ead of swap
change = 10000
End If
If fneigh < fbest Then
solbest = solneigh: fbest = fneigh: Cmaxbest = Cmaxneigh: Tmaxbest =
Tmaxneigh
End If
Loop

endTime = Timer
Time = endTime - startTime

'printing information
display2.Cls
display2.Print "--------------------------------------------------------------
-------------------------
-----------------"
display2.Print " Current Sol", "Best Sol", "Cmax", "Tmax", " iterations", "
Time(s)"
display2.Print fcurr, fbest, Cmaxbest, Tmaxbest, it, Time
display2.Print
display2.Print " Solution Sequence"
display2.Print "--------------------------------------------------------------
-------------------------
-----------------"
display2.Print " Machine 1", " Machine 2", " Machine 3", " Machine 4"

For i = 1 To n
   For k = 1 To m
   display2.Print solbest(k, i),
   Next k
   display2.Print
Next i

'printing info to archive
Print #1, "General info about problem and the parameters"
Print #1, "-------------------------------------------------------------------
-------------------------
---"
Print #1, "n", "m", "ex", "p1", "p2", "p3", "p4", "p5", "finit",
"fbesthalf", "fbest", "Cmaxbest",
"Tmaxbest", "Time(s)", "it", "filepath"
```

```
Print #1, n, m, ex, p1, p2, p3, p4, p5, finit, fbesthalf, fbest, Cmaxbest,
Tmaxbest, Time, it, file
name
Print #1, "------------------------------------------------------------
-----------------------
-----------"
Print #1, " Current Sol", "Best Sol", "Cmax", "Tmax", " iterations", "
Time(s)"
Print #1, fcurr, fbest, Cmaxbest, Tmaxbest, it, Time
Print #1,
Print #1, " Solution Sequence"
Print #1, "------------------------------------------------------------
-----------------------
-----------"
Print #1, " Machine 1", " Machine 2", " Machine 3", " Machine 4"
For i = 1 To n
   For k = 1 To m
   Print #1, solbest(k, i),
   Next k
   Print #1,
Next i
End Sub

'list data after SPT criteria
Private Sub orderSPT()

   Dim tempd As Single, tempp As Single, tempj As Single

       For i = 1 To n - 1
           For k = i + 1 To n
           If p(k) < p(i) Then
               tempd = d(i): tempp = p(i): tempj = j(i)
               d(i) = d(k): p(i) = p(k): j(i) = j(k)
               d(k) = tempd: p(k) = tempp: j(k) = tempj
           End If
       Next k
       Next i

End Sub

'list data after EDD criteria
Private Sub orderEDD()

   Dim tempd As Single, tempp As Single, tempj As Single

       For i = 1 To n - 1
           For k = i + 1 To n
           If d(k) < d(i) Then
               tempd = d(i): tempp = p(i): tempj = j(i)
               d(i) = d(k): p(i) = p(k): j(i) = j(k)
               d(k) = tempd: p(k) = tempp: j(k) = tempj
           End If
       Next k
       Next i
End Sub

'list data after SPT + EDD criteria
Private Sub ordermixed()

   Dim tempd As Single, tempp As Single, tempj As Single, mixed() As
Integer
   ReDim mixed(n)
   For i = 1 To n
       mixed(i) = 0.2 * p(i) + 0.8 * d(i)
   Next i

       For i = 1 To n - 1
           For k = i + 1 To n
           If mixed(k) < mixed(i) Then
               tempd = d(i): tempp = p(i): tempj = j(i)
               d(i) = d(k): p(i) = p(k): j(i) = j(k)
               d(k) = tempd: p(k) = tempp: j(k) = tempj
           End If
       Next k
       Next i
End Sub

'initial asignment of jobs to machines and results calculation
'the jobs are asigned to the machine where the completion time will be the
minimum posible
Private Sub asign()
Cmax = 0: Tmax = 0
ReDim tempC(m): ReDim Counter(m): ReDim Cseq(m)
ReDim solinit(m, n): ReDim solcurr(m, n): ReDim solneigh(m, n): ReDim
solbest(m, n)
ReDim c(n): ReDim L(n): ReDim T(n)
For i = 1 To n

   min = 10000

   For k = 1 To m
   tempC(k) = Cseq(k) + s(solcurr(k, Counter(k)), j(i)) + p(i)
'calculates completion time for the job in all machines

'finds the machine with minimum completion time for the job
       If tempC(k) < min Then
       min = tempC(k): Index = k
       End If
   Next k
   k = Index

   Cseq(k) = tempC(k)                          'completion time for machine
   Counter(k) = Counter(k) + 1                 'adds 1 if the job is
asigned to machine
   solcurr(k, Counter(k)) = j(i)               'adds the job to the
sequence
   c(i) = Cseq(k)                              'saves the completion time
for the job
   L(i) = c(i) - d(i)                          'calculate the lateness for
the job

   If L(i) < 0 Then                            'calculates the tardiness
for the job
   T(i) = 0
   Else
   T(i) = L(i)
   End If

   If c(i) > Cmax Then
   Cmax = c(i)
   End If

   If T(i) > Tmax Then
   Tmax = T(i)
   End If

Next i
fcurr = alfa * Cmax + (1 - alfa) * Tmax
'printing info
display1.Cls
display1.Print " F", " Cmax", "Tmax"
display1.Print "------------------------------------------------------------
------------------------
--------"
display1.Print fcurr, Cmax, Tmax
display1.Print ""
display1.Print " Sequence"
display1.Print "------------------------------------------------------------
------------------------
--------"
display1.Print " Machine 1", " Machine 2", " Machine 3", " Machine 4"
For i = 1 To n
   For k = 1 To m
   display1.Print solcurr(k, i),
   Next k
   display1.Print
Next i
display2.Cls
'printing info to archive
Print #1, "Initial Solution"
Print #1, "-----------------------------------------------------------------
------------------------
---"
Print #1, " F", " Cmax", "Tmax"
Print #1, "-----------------------------------------------------------------
------------------------
---"
Print #1, fcurr, Cmax, Tmax
Print #1,
Print #1, " Sequence"
Print #1, "-----------------------------------------------------------------
------------------------
---"
Print #1, " Machine 1", " Machine 2", " Machine 3", " Machine 4"
For i = 1 To n
   For k = 1 To m
   Print #1, solcurr(k, i),
   Next k
   Print #1,
Next i
Print #1, "-----------------------------------------------------------------
------------------------
---"
Cmaxbest = Cmax: Tmaxbest = Tmax
Cmax = 0: Tmax = 0
solinit = solcurr: finit = fcurr                    'saves the initial
solution values
fbest = fcurr: solbest = solcurr                    'saves the initial
solution as the current solution and the best solution
End Sub

Private Sub asignrdm()
ReDim solinit(m, n): ReDim solcurr(m, n): ReDim solneigh(m, n): ReDim
solbest(m, n)
ReDim c(n): ReDim L(n): ReDim T(n): ReDim Cseq(m)
```

```
Dim tempj As Single, replace As Single
Dim Counter As Single
Cmax = 0: Tmax = 0
For i = 1 To n
   replace = Round(Rnd * n + 0.5)
   tempj = j(i)
   j(i) = j(replace)
   j(replace) = tempj
Next i
Counter = 1: k = 1: i = 1
   Do While i < n + 1
   solneigh(k, Counter) = j(i)
   k = k + 1: i = i + 1
   If k > m Then
   k = 1: Counter = Counter + 1
   End If
   Loop
Call calc
solcurr = solneigh
fcurr = fneigh
solinit = solcurr: finit = fcurr                    'saves the initial
solution values
fbest = fcurr: solbest = solcurr                    'saves the initial
solution as the current solution and the best solution
Cmaxbest = Cmaxneigh: Tmaxbest = Tmaxneigh
'printing info
display1.Cls
display1.Print " F", " Cmax", "Tmax"
display1.Print "--------------------------------------------------------------------
------------------------
--------"
display1.Print fcurr, Cmaxbest, Tmaxbest
display1.Print ""
display1.Print " Sequence"
display1.Print "--------------------------------------------------------------------
------------------------
--------"
display1.Print " Machine 1", " Machine 2", " Machine 3", " Machine 4"
For i = 1 To n
   For k = 1 To m
   display1.Print solcurr(k, i),
   Next k
   display1.Print
Next i
display2.Cls
'printing info to archive
Print #1, "Initial Solution"
Print #1, "--------------------------------------------------------------------
------------------------
---"
Print #1, " F", " Cmax", "Tmax"
Print #1, "--------------------------------------------------------------------
------------------------
---"
Print #1, fcurr, Cmaxbest, Tmaxbest
Print #1,
Print #1, " Sequence"
Print #1, "--------------------------------------------------------------------
------------------------
---"
Print #1, " Machine 1", " Machine 2", " Machine 3", " Machine 4"
For i = 1 To n
```

```
   For k = 1 To m
   Print #1, solcurr(k, i),
   Next k
   Print #1,
Next i
Print #1, "-----------------------------------------------------------------
-----------------------
---"
End Sub

' calculates target criterias for any given solution
Private Sub calc()
For z = 1 To m

   Cseq(z) = 0

   For i = 1 To n

   Cseq(z) = Cseq(z) + s(solneigh(z, i - 1), solneigh(z, i)) +
p(solneigh(z, i))      'completion
time for each job in each machine

       If solneigh(z, i) = 0 Then
'if there are no more jobs to sequence they cannot have completion time
       c(solneigh(z, i)) = 0
       Else
       c(solneigh(z, i)) = Cseq(z)
       End If

   L(solneigh(z, i)) = c(solneigh(z, i)) - d(solneigh(z, i))

       If L(solneigh(z, i)) < 0 Then
       T(solneigh(z, i)) = 0
       Else
       T(solneigh(z, i)) = L(solneigh(z, i))
       End If

If c(solneigh(z, i)) >= Cmax Then
Cmax = c(solneigh(z, i))
End If
If T(solneigh(z, i)) >= Tmax Then
Tmax = T(solneigh(z, i))
End If
   Next i
Next z
fneigh = alfa * Cmax + (1 - alfa) * Tmax           'calculates target value
Cmaxneigh = Cmax: Tmaxneigh = Tmax
Cmax = 0: Tmax = 0
End Sub

Private Sub swap()
solneigh = solcurr
x = Round(Rnd() * n + 0.5, 0)                       'selects randomly two
different jobs between job 1 to n
y = Round(Rnd() * n + 0.5, 0)
   Do While x = y
   y = Round(Rnd() * n + 0.5, 0)
   Loop
For k = 1 To m                                      'interchanges the
position of the selected jobs
   For i = 1 To n
```

```
        If solcurr(k, i) = x Then
        solneigh(k, i) = y
        End If

        If solcurr(k, i) = y Then
        solneigh(k, i) = x
        End If

   Next i
Next k

Call calc
End Sub

Private Sub insert()
ReDim soltemp(m, n)
solneigh = solcurr
x = Round(Rnd() * n + 0.5, 0)                        'selects randomly one
job
y = Round(Rnd() * m + 0.5, 0)                        'selects randomly the
machine to which the job is moved
For k = 1 To m                                       'finds the position of
the selected job
   For i = 1 To n

        If solcurr(k, i) = x Then
        mpos = k: ppos = i
        End If

   Next i
Next k
z = 0
For i = 1 To n - 2                                 'finds out the number of
jobs sequenced in the machine
   If solcurr(y, i) <> 0 Then
   z = z + 1
   End If

Next i

If y = mpos Then
v = Round(Rnd() * z + 0.5, 0)                     'selects the position
where the job is inserted
   For i = ppos To n - 2                            'removes the selected
job from its position
   solneigh(mpos, i) = solcurr(mpos, i + 1)
   Next i

   soltemp = solneigh

   solneigh(y, v) = x                             'inserts the selected job in
the new position
   For i = v + 1 To n - 2                         'reestablishes the rest of
the sequence
        solneigh(y, i) = soltemp(y, i - 1)
   Next i
Else
z = z + 1

v = Round(Rnd() * z + 0.5, 0)                     'selects the position
where the job is inserted
   solneigh(y, v) = x                              'inserts the job in the
```

```
position and pushes the rest of the jobs one step forward in the sequence
   For i = v + 1 To n - 2
       solneigh(y, i) = solcurr(y, i - 1)
   Next i
   For i = ppos To n - 2                         'removes the selected job
from its original sequence
       solneigh(mpos, i) = solcurr(mpos, i + 1)
   Next i
End If

Call calc
mpos = 0: ppos = 0
End Sub

Private Sub mixed_Click()
initmethod = 3
EDD.Checked = False
SPT.Checked = False
mixed.Checked = True
random.Checked = False
If SAS.Checked = True Or SAI.Checked = True Or SASI.Checked = True Then
solve.Enabled = True
End If
End Sub

Private Sub SPT_Click()
initmethod = 2
EDD.Checked = False
SPT.Checked = True
mixed.Checked = False
random.Checked = False
If SAS.Checked = True Or SAI.Checked = True Or SASI.Checked = True Then
solve.Enabled = True
End If
End Sub
```

Appendix D

In this appendix some examples of output data are given from each of the phases of the computational experimentation. For the complete results from the computational experimentation, see the CD. For explanations to abbreviations, see Chapter 1 Abbreviations and Symbols.

Pre-Phase 1

For the complete results, see the archive Pre-phase 1.xls on the CD.
The example given here is the output data of the RSAS algorithm, with 20 examples for each configuration of jobs and machines.

n	m	ex	p1	p2	p3	p4	p5	finit	fbest	Cmaxbest	Tmaxbest	Time	it
15	2	1	100	1,2	0,8	0,8	10	650	345	564	199	5,328125	217948
15	2	2	100	1,2	0,8	0,8	10	530,2	312,2	479	201	3,59375	146397
15	2	3	100	1,2	0,8	0,8	10	488,2	313,6	460	216	3,4375	140326
15	2	4	100	1,2	0,8	0,8	10	653,4	366,6	654	175	4,046875	164908
15	2	5	100	1,2	0,8	0,8	10	659	386	515	300	5,8125	238212
15	2	6	100	1,2	0,8	0,8	10	633,6	303,6	549	140	3,78125	154441
15	2	7	100	1,2	0,8	0,8	10	541,8	285,4	484	153	3,0625	124112
15	2	8	100	1,2	0,8	0,8	10	656,8	304,6	553	139	3,59375	146875
15	2	9	100	1,2	0,8	0,8	10	657	389,2	535	292	4,359375	178292
15	2	10	100	1,2	0,8	0,8	10	575	286,2	504	141	4,921875	200684
15	2	1	100	0,5	0,2	0,2	10	209,2	170,8	427	0	4,65625	190643
15	2	2	100	0,5	0,2	0,2	10	397,6	190	475	0	5,5	225930
15	2	3	100	0,5	0,2	0,2	10	306	177,2	443	0	4,703125	192441
15	2	4	100	0,5	0,2	0,2	10	200,6	132,4	331	0	5,921875	239864
15	2	5	100	0,5	0,2	0,2	10	274,8	190,4	476	0	5,375	220898
15	2	6	100	0,5	0,2	0,2	10	329,8	223,8	522	25	4,15625	170620
15	2	7	100	0,5	0,2	0,2	10	302,6	171,6	429	0	4,234375	173744
15	2	8	100	0,5	0,2	0,2	10	233,8	172,4	431	0	4,828125	198027
15	2	9	100	0,5	0,2	0,2	10	291	171,6	429	0	4,640625	190255
15	2	10	100	0,5	0,2	0,2	10	263	180	450	0	5,03125	206441
20	2	1	100	0,5	0,4	0,8	10	385	204,8	512	0	6,75	215430
20	2	2	100	0,5	0,4	0,8	10	596,6	315,6	672	78	5,390625	171182
20	2	3	100	0,5	0,4	0,8	10	383,6	192,4	481	0	6,5625	209255
20	2	4	100	0,5	0,4	0,8	10	423,8	233,6	584	0	6,75	215055
20	2	5	100	0,5	0,4	0,8	10	546,2	266,8	667	0	5,921875	188765
20	2	6	100	0,5	0,4	0,8	10	525	252	630	0	6,515625	208063
20	2	7	100	0,5	0,4	0,8	10	582,2	250,4	626	0	5,875	187386
20	2	8	100	0,5	0,4	0,8	10	565,8	220,4	551	0	5,875	187447
20	2	9	100	0,5	0,4	0,8	10	420,4	216	540	0	6,34375	201988
20	2	10	100	0,5	0,4	0,8	10	487,2	216	540	0	6,828125	217998
20	2	1	100	0,5	0,6	0,2	10	419,2	292	511	146	6,0625	193324
20	2	2	100	0,5	0,6	0,2	10	503,2	330,8	548	186	6,5625	208158
20	2	3	100	0,5	0,6	0,2	10	568,4	429,2	644	286	5,21875	165685
20	2	4	100	0,5	0,6	0,2	10	534,4	366,8	584	222	5,453125	173468
20	2	5	100	0,5	0,6	0,2	10	377,2	267,6	483	124	6,234375	198604
20	2	6	100	0,5	0,6	0,2	10	478	350,2	562	209	5,171875	164514
20	2	7	100	0,5	0,6	0,2	10	482,6	327,8	545	183	5,90625	187688
20	2	8	100	0,5	0,6	0,2	10	435	306,6	525	161	4,90625	155895
20	2	9	100	0,5	0,6	0,2	10	642	454,6	673	309	7,0625	224302
20	2	10	100	0,5	0,6	0,2	10	498,8	376,8	579	242	6,1875	196987
n	m	ex	p1	p2	p3	p4	p5	finit	fbest	Cmaxbest	Tmaxbest	Time	it
20	3	1	100	1,2	0,8	0,8	10	562,2	344	491	246	7,546875	168649

20	3	2	100	1,2	0,8	0,8	10	423	245,4	354	173	11,92188	266320
20	3	3	100	1,2	0,8	0,8	10	615,4	315,8	512	185	8,21875	183506
20	3	4	100	1,2	0,8	0,8	10	548,2	244	454	104	8,203125	183378
20	3	5	100	1,2	0,8	0,8	10	478,6	229	406	111	7,921875	176675
20	3	6	100	1,2	0,8	0,8	10	507,4	274	442	162	7,53125	168149
20	3	7	100	1,2	0,8	0,8	10	480,6	284,6	428	189	7,03125	157192
20	3	8	100	1,2	0,8	0,8	10	543	251,8	400	153	9,71875	217342
20	3	9	100	1,2	0,8	0,8	10	605,8	302	437	212	9,671875	216104
20	3	10	100	1,2	0,8	0,8	10	427,2	218,8	367	120	8,171875	182344
20	3	1	100	0,5	0,2	0,2	10	233,6	122,8	307	0	9,078125	199650
20	3	2	100	0,5	0,2	0,2	10	152,8	126,4	316	0	9,15625	201120
20	3	3	100	0,5	0,2	0,2	10	260	168,4	421	0	7,8125	174726
20	3	4	100	0,5	0,2	0,2	10	173,4	151,2	378	0	11,60938	257764
20	3	5	100	0,5	0,2	0,2	10	199,8	149,6	374	0	11,65625	258652
20	3	6	100	0,5	0,2	0,2	10	263,4	164,4	411	0	10,04688	224399
20	3	7	100	0,5	0,2	0,2	10	247,8	166,4	416	0	9,578125	214073
20	3	8	100	0,5	0,2	0,2	10	373,4	182,8	448	6	8,265625	185434
20	3	9	100	0,5	0,2	0,2	10	213,6	142,4	356	0	9,4375	209381
20	3	10	100	0,5	0,2	0,2	10	226,4	146	365	0	9,90625	219734
50	3	1	100	1,2	0,8	0,8	10	1146,4	563,4	1032	251	18,5	173777
50	3	2	100	1,2	0,8	0,8	10	1131,6	572,8	1072	240	17,4375	163707
50	3	3	100	1,2	0,8	0,8	10	1398	710,2	1117	439	18,54688	174145
50	3	4	100	1,2	0,8	0,8	10	1232	666	1056	406	17,07813	160438
50	3	5	100	1,2	0,8	0,8	10	1117,6	591,2	1031	298	19,75	185440
50	3	6	100	1,2	0,8	0,8	10	1186	612,2	1025	337	18,07813	169873
50	3	7	100	1,2	0,8	0,8	10	1089	486,6	972	163	18,73438	176139
50	3	8	100	1,2	0,8	0,8	10	1104	567,4	973	297	19,70313	185178
50	3	9	100	1,2	0,8	0,8	10	1170,8	615,4	1036	335	17,4375	163765
50	3	10	100	1,2	0,8	0,8	10	1034,2	573	948	323	19,71875	185226
50	3	1	100	0,5	0,2	0,2	10	469,6	374,4	936	0	27,40625	253384
50	3	2	100	0,5	0,2	0,2	10	406,6	340,4	851	0	22,8125	209803
50	3	3	100	0,5	0,2	0,2	10	472	367,6	919	0	21,03125	194989
50	3	4	100	0,5	0,2	0,2	10	529,4	365,6	914	0	21,29688	197057
50	3	5	100	0,5	0,2	0,2	10	379,6	326,8	817	0	21,17188	194242
50	3	6	100	0,5	0,2	0,2	10	433	316	790	0	21,46875	197032
50	3	7	100	0,5	0,2	0,2	10	526,8	360,4	901	0	26,79688	247615
50	3	8	100	0,5	0,2	0,2	10	505,4	364	910	0	25,45313	235097
50	3	9	100	0,5	0,2	0,2	10	500,4	380,8	952	0	29,14063	269718
50	3	10	100	0,5	0,2	0,2	10	466,8	348,8	872	0	21,60938	199219
50	4	1	100	0,5	0,8	0,8	10	899,2	503,4	759	333	28,625	204774
50	4	2	100	0,5	0,8	0,8	10	777	346,6	607	173	27,18701	194590
50	4	3	100	0,5	0,8	0,8	10	699	413,6	668	244	24,03101	172007
50	4	4	100	0,5	0,8	0,8	10	675,4	399,4	643	237	24,75	176981
50	4	5	100	0,5	0,8	0,8	10	845,4	409,6	688	224	25,06299	179162
50	4	6	100	0,5	0,8	0,8	10	753	413,4	675	239	26,01611	185977
50	4	7	100	0,5	0,8	0,8	10	799,2	393,2	698	190	26,23389	187550
50	4	8	100	0,5	0,8	0,8	10	854,6	436,6	709	255	26,06201	186464
50	4	9	100	0,5	0,8	0,8	10	769,8	448,6	730	261	25,04712	179234
50	4	10	100	0,5	0,8	0,8	10	803,8	442,2	732	249	27,03101	193373
50	4	1	100	0,8	0,2	0,2	10	347	274,4	686	0	30,875	214148
50	4	2	100	0,8	0,2	0,2	10	492,4	310	775	0	27,04688	189751
50	4	3	100	0,8	0,2	0,2	10	320,8	254,4	636	0	28,57813	198135
50	4	4	100	0,8	0,2	0,2	10	418,4	298,4	746	0	30,90601	215524
50	4	5	100	0,8	0,2	0,2	10	501,4	281,6	704	0	31,25	217101
50	4	6	100	0,8	0,2	0,2	10	288	251,6	629	0	31,79688	220555
50	4	7	100	0,8	0,2	0,2	10	356	296,8	742	0	38,28101	266322
50	4	8	100	0,8	0,2	0,2	10	335,2	275,2	688	0	30,65601	212792
50	4	9	100	0,8	0,2	0,2	10	360,4	287,2	718	0	33,81299	235205
50	4	10	100	0,8	0,2	0,2	10	337,2	266,8	667	0	33,5459	232857

Pre-phase 2

For the complete results, see the archive Pre-phase 2.xls on the CD.

The summary is given together with the calculations from 100 instances as an example.

SPT			EDD			20% SPT / 80 % EDD			Random		
Jobs	Ie	Var	Jobs	Ie	Var	Jobs	Ie	Var	Jobs	Ie	Var
15	29,1%	0,0274	15	1,0%	0,0003	15	2,2%	0,0011	15	36,5%	0,0369
20	32,1%	0,0326	20	1,1%	0,0003	20	1,9%	0,0009	20	42,9%	0,0521
50	46,3%	0,0446	50	0,5%	0,0001	50	1,7%	0,0004	50	56,8%	0,0557
Overall	37,2%	0,0419	Overall	0,9%	0,0003	Overall	1,9%	0,0007	Overall	47,2%	0,0573

finit	Ie	finit	Ie	finit	Ie	finit	Ie	Minimum
553,6	16,7%	491,6	3,6%	474,4	0,0%	603,8	27,3%	474,4
472,2	11,9%	422	0,0%	460	9,0%	530,8	25,8%	422
490,2	22,3%	416,8	4,0%	400,8	0,0%	481,4	20,1%	400,8
686,8	17,7%	583,4	0,0%	629,4	7,9%	595	2,0%	583,4
496	0,5%	509,8	3,3%	493,4	0,0%	542,2	9,9%	493,4
530	5,4%	503	0,0%	544	8,2%	609,6	21,2%	503
435	0,0%	456,4	4,9%	436,4	0,3%	550,2	26,5%	435
504,8	5,7%	477,4	0,0%	516,8	8,3%	573	20,0%	477,4
626,2	30,0%	481,8	0,0%	519,8	7,9%	681,8	41,5%	481,8
482	4,8%	481	4,6%	460	0,0%	535	16,3%	460
572,8	23,8%	462,8	0,0%	465,4	0,6%	603,2	30,3%	462,8
486,6	26,1%	386	0,0%	410	6,2%	510,4	32,2%	386
513	38,9%	371,2	0,5%	369,2	0,0%	532,8	44,3%	369,2
683,2	34,4%	535,4	5,3%	508,4	0,0%	620,4	22,0%	508,4
528,4	19,2%	470,2	6,1%	443,2	0,0%	596	34,5%	443,2
503,6	10,0%	458	0,0%	499	9,0%	678,4	48,1%	458
424,2	0,0%	429,4	1,2%	425,4	0,3%	451,2	6,4%	424,2
501,8	18,0%	432,4	1,6%	425,4	0,0%	563	32,3%	425,4
673	49,0%	451,8	0,0%	451,8	0,0%	596,8	32,1%	451,8
466,4	12,0%	435,6	4,6%	416,6	0,0%	586,8	40,9%	416,6
591,4	36,3%	434	0,0%	435,4	0,3%	613	41,2%	434
490,2	28,1%	382,6	0,0%	402,8	5,3%	484,2	26,6%	382,6
484,2	36,2%	355,4	0,0%	374	5,2%	513,8	44,6%	355,4
679,6	39,4%	487,4	0,0%	487,4	0,0%	824	69,1%	487,4
585	28,3%	455,8	0,0%	456,4	0,1%	686	50,5%	455,8
492,2	19,0%	413,6	0,0%	454,6	9,9%	660,4	59,7%	413,6
413,4	3,9%	401,8	1,0%	397,8	0,0%	554	39,3%	397,8
521	37,0%	387,4	1,8%	380,4	0,0%	552,6	45,3%	380,4
702	59,4%	440,4	0,0%	440,4	0,0%	572,2	29,9%	440,4
497	28,2%	387,6	0,0%	388,6	0,3%	576,8	48,8%	387,6
626,6	61,5%	388	0,0%	437,2	12,7%	657	69,3%	388
490,2	32,1%	371,2	0,0%	379	2,1%	520,8	40,3%	371,2
507	41,5%	358,4	0,0%	375,2	4,7%	596,2	66,4%	358,4
675,4	53,7%	439,4	0,0%	439,4	0,0%	649,8	47,9%	439,4
586,8	25,6%	467,2	0,0%	467,2	0,0%	693	48,3%	467,2
496	30,3%	380,6	0,0%	421,6	10,8%	658,8	73,1%	380,6
402,6	11,9%	374,8	4,2%	359,8	0,0%	454,4	26,3%	359,8
539,6	54,9%	348,4	0,0%	351,6	0,9%	608,4	74,6%	348,4
670	45,2%	461,4	0,0%	461,4	0,0%	681	47,6%	461,4
485,2	39,9%	346,8	0,0%	346,8	0,0%	534,8	54,2%	346,8
486,8	7,1%	461,4	1,5%	454,4	0,0%	590,2	29,9%	454,4
431,2	14,6%	376,4	0,0%	396,4	5,3%	433,8	15,2%	376,4
508,6	8,0%	470,8	0,0%	473,8	0,6%	471	0,0%	470,8
418	14,5%	365,2	0,0%	374,2	2,5%	402,4	10,2%	365,2
527,6	17,2%	488	8,4%	495	10,0%	450,2	0,0%	450,2
471,6	12,0%	421,2	0,0%	427,8	1,6%	546	29,6%	421,2
528	13,5%	468,2	0,6%	465,2	0,0%	570,4	22,6%	465,2
462,8	17,9%	410,4	4,6%	392,4	0,0%	541,6	38,0%	392,4
518	15,2%	463,2	3,0%	449,6	0,0%	477,6	6,2%	449,6
462,6	3,4%	447,6	0,0%	457,6	2,2%	484,2	8,2%	447,6
497,4	14,5%	434,4	0,0%	434,4	0,0%	493,8	13,7%	434,4

finit	Ie	finit	Ie	finit	Ie	finit	Ie	Minimum
461	31,7%	350	0,0%	362,6	3,6%	412,6	17,9%	350
532,4	22,9%	433,2	0,0%	435,8	0,6%	512,4	18,3%	433,2
443,2	33,7%	331,6	0,0%	334,6	0,9%	391,4	18,0%	331,6
543,8	23,5%	465,2	5,7%	440,2	0,0%	518,6	17,8%	440,2
504,6	27,2%	396,6	0,0%	398,2	0,4%	456,6	15,1%	396,6
525,6	20,9%	437,6	0,7%	434,6	0,0%	511,4	17,7%	434,6
450,8	19,8%	383,4	1,9%	376,4	0,0%	406,4	8,0%	376,4
549,2	23,9%	443,4	0,0%	448,6	1,2%	619,2	39,6%	443,4
431,4	3,6%	416,4	0,0%	440,4	5,8%	562	35,0%	416,4
514	31,2%	391,8	0,0%	391,8	0,0%	577	47,3%	391,8
466	46,4%	318,2	0,0%	332,8	4,6%	501	57,4%	318,2
540,2	33,2%	405,6	0,0%	408,2	0,6%	497,4	22,6%	405,6
447	49,9%	298,2	0,0%	326,8	9,6%	414	38,8%	298,2
560,6	35,0%	420,4	1,2%	415,4	0,0%	563	35,5%	415,4
508,8	36,8%	372	0,0%	373,6	0,4%	612	64,5%	372
524,2	29,8%	407,6	0,9%	404	0,0%	565,6	40,0%	404
432,2	22,3%	353,4	0,0%	354,4	0,3%	533	50,8%	353,4
563	25,5%	448,6	0,0%	457,8	2,1%	537	19,7%	448,6
406,8	5,8%	384,6	0,0%	408,6	6,2%	543,6	41,3%	384,6
514	32,6%	387,6	0,0%	387,6	0,0%	520,2	34,2%	387,6
466	44,0%	323,6	0,0%	323,6	0,0%	428	32,3%	323,6
547,4	44,8%	378	0,0%	386	2,1%	512,8	35,7%	378
447	49,9%	298,2	0,0%	334,6	12,2%	424,6	42,4%	298,2
576,8	45,1%	397,6	0,0%	420,6	5,8%	576,6	45,0%	397,6
508,8	46,5%	347,4	0,0%	355,6	2,4%	519	49,4%	347,4
521,8	38,0%	381,2	0,8%	378,2	0,0%	553	46,2%	378,2
424,6	31,7%	322,4	0,0%	331,4	2,8%	541	67,8%	322,4
541	21,3%	446	0,0%	446	0,0%	544	22,0%	446
388,2	10,0%	352,8	0,0%	376,8	6,8%	502,8	42,5%	352,8
516,8	10,5%	485,8	3,8%	467,8	0,0%	474,8	1,5%	467,8
439,2	14,7%	390	1,9%	382,8	0,0%	440	14,9%	382,8
463,2	19,2%	411,6	5,9%	388,6	0,0%	500	28,7%	388,6
536,8	16,6%	469,4	2,0%	460,4	0,0%	497,8	8,1%	460,4
401,8	20,4%	333,6	0,0%	344,2	3,2%	404,8	21,3%	333,6
439,8	19,8%	369,2	0,5%	367,2	0,0%	430,8	17,3%	367,2
411,6	11,7%	368,6	0,0%	375,6	1,9%	471,6	27,9%	368,6
460,6	33,1%	346	0,0%	348,6	0,8%	462,2	33,6%	346
502,2	17,4%	427,6	0,0%	441,2	3,2%	460	7,6%	427,6
329,4	6,3%	310	0,0%	328,4	5,9%	356	14,8%	310
549,2	24,0%	454	2,5%	442,8	0,0%	648,6	46,5%	442,8
459,6	27,7%	366	1,7%	359,8	0,0%	488,6	35,8%	359,8
475,2	30,4%	374,4	2,7%	364,4	0,0%	470,4	29,1%	364,4
557,8	29,2%	431,6	0,0%	440,6	2,1%	533,8	23,7%	431,6
405,4	29,7%	312,6	0,0%	318,8	2,0%	449,2	43,7%	312,6
454,8	37,7%	330,2	0,0%	330,2	0,0%	423,2	28,2%	330,2
432	26,9%	340,4	0,0%	343,4	0,9%	487,8	43,3%	340,4
490	57,2%	311,8	0,0%	311,8	0,0%	559,2	79,3%	311,8
540	31,3%	411,4	0,0%	434,6	5,6%	525	27,6%	411,4
294	7,8%	272,8	0,0%	272,8	0,0%	332,8	22,0%	272,8

Phase 1: Tuning

For the complete results, see the archive Phase 1 Tuning.xls on the CD.

For both RSAS and RSAI the summary is given as well as the calculations from 100 instances for each of the 4 experiments realized.

Jobs	Ie	Var	Time	Ie	Var	Time	Ie	Var	Time	Ie	Var	Time
20	0,81%	0,00008	15,8	0,97%	0,00009	7,9	0,92%	0,00010	15,1	0,82%	0,00009	7,6
50	0,65%	0,00032	43,1	1,07%	0,00009	24,1	0,69%	0,00006	43,8	0,95%	0,00008	24,6

	RSAS							
MIN	Ex1	Ie	Ex2	Ie	Ex3	Ie	Ex4	Ie
349	353	1,1%	349	0,0%	350,8	0,5%	350	0,3%
440,4	440,4	0,0%	443	0,6%	448,8	1,9%	444,8	1,0%
372,4	372,4	0,0%	377,2	1,3%	378,8	1,7%	376,2	1,0%
402,6	403,6	0,2%	409,6	1,7%	402,8	0,0%	402,6	0,0%
455	463,2	1,8%	457	0,4%	455	0,0%	456,2	0,3%
332	334,2	0,7%	339,6	2,3%	332	0,0%	336,2	1,3%
478,6	482	0,7%	487,8	1,9%	481,6	0,6%	478,6	0,0%
369	378	2,4%	369	0,0%	378	2,4%	385,2	4,4%
426,4	432,8	1,5%	431,4	1,2%	426,4	0,0%	430,2	0,9%
455,2	460,4	1,1%	463,2	1,8%	455,4	0,0%	455,2	0,0%
324,8	327,4	0,8%	325,8	0,3%	324,8	0,0%	325,2	0,1%
400,2	400,2	0,0%	401,6	0,3%	401,6	0,3%	405	1,2%
337	340	0,9%	340,4	1,0%	337	0,0%	340	0,9%
366,2	366,2	0,0%	372,8	1,8%	373,8	2,1%	366,4	0,1%
411,8	424,6	3,1%	417	1,3%	411,8	0,0%	420,2	2,0%
298,4	298,6	0,1%	298,4	0,0%	303,2	1,6%	305,4	2,3%
426,6	437,6	2,6%	426,6	0,0%	434,6	1,9%	436,2	2,3%
360,8	364	0,9%	360,8	0,0%	364,2	0,9%	364,2	0,9%
421	423	0,5%	421	0,0%	421	0,0%	422,2	0,3%
446	452	1,3%	460,4	3,2%	446	0,0%	455,8	2,2%
321,4	322,2	0,2%	321,4	0,0%	328,8	2,3%	326,6	1,6%
393,8	400,4	1,7%	401,4	1,9%	393,8	0,0%	402	2,1%
309,8	316,8	2,3%	316,6	2,2%	309,8	0,0%	309,8	0,0%
371,2	373,2	0,5%	371,2	0,0%	371,4	0,1%	372,2	0,3%
411,4	411,4	0,0%	415,6	1,0%	412,6	0,3%	416,6	1,3%
291,2	291,2	0,0%	294	1,0%	292,4	0,4%	296	1,6%
395,2	399,6	1,1%	401	1,5%	396	0,2%	395,2	0,0%
376,8	376,8	0,0%	379,6	0,7%	380,4	1,0%	380,6	1,0%
412,2	423,4	2,7%	423,4	2,7%	421,4	2,2%	412,2	0,0%
449,4	449,4	0,0%	459,8	2,3%	450,6	0,3%	455,2	1,3%
326	326	0,0%	331,4	1,7%	329,2	1,0%	332,8	2,1%
406,2	409,8	0,9%	411,8	1,4%	407	0,2%	406,2	0,0%
305,8	310,8	1,6%	305,8	0,0%	314	2,7%	310,8	1,6%
365,2	367,2	0,5%	369	1,0%	367,4	0,6%	365,2	0,0%
433,4	440,2	1,6%	433,4	0,0%	439,2	1,3%	441,4	1,8%
309,4	309,4	0,0%	314,6	1,7%	312,6	1,0%	312,8	1,1%
393	402,2	2,3%	393	0,0%	394,2	0,3%	395	0,5%
394,2	398,4	1,1%	396,4	0,6%	397,4	0,8%	394,2	0,0%
413,2	423	2,4%	420,2	1,7%	420,2	1,7%	413,2	0,0%
448,6	448,6	0,0%	457	1,9%	450	0,3%	451	0,5%
437	437	0,0%	440,8	0,9%	442	1,1%	437,8	0,2%
369	374	1,4%	375,8	1,8%	369,6	0,2%	369	0,0%
445,4	446,8	0,3%	449	0,8%	448,2	0,6%	445,4	0,0%
464	468,4	0,9%	464	0,0%	471,2	1,6%	471,4	1,6%
356,2	362,2	1,7%	362,2	1,7%	358,2	0,6%	356,2	0,0%
531,6	538,2	1,2%	533,2	0,3%	535,6	0,8%	531,6	0,0%
491,2	497,6	1,3%	495,2	0,8%	494,8	0,7%	491,2	0,0%
406,6	406,6	0,0%	407	0,1%	410	0,8%	407	0,1%
475,8	484,8	1,9%	484	1,7%	483,2	1,6%	475,8	0,0%
489,8	492	0,4%	494,4	0,9%	489,8	0,0%	493	0,7%
418,6	423,6	1,2%	420	0,3%	418,6	0,0%	420	0,3%

MIN	Ex1	Ie	Ex2	Ie	Ex3	Ie	Ex4	Ie
337,2	341	1,1%	337,2	0,0%	343	1,7%	339,8	0,8%
436,2	438,6	0,6%	437,4	0,3%	436,2	0,0%	440,4	1,0%
438,2	438,2	0,0%	440,4	0,5%	442,8	1,0%	440,4	0,5%
325,6	331	1,7%	325,6	0,0%	328	0,7%	325,8	0,1%
498,2	500,4	0,4%	507,4	1,8%	500,2	0,4%	498,2	0,0%
469	469	0,0%	469,4	0,1%	469,2	0,0%	471,6	0,6%
386,4	388,4	0,5%	390	0,9%	394	2,0%	386,4	0,0%
441,6	441,6	0,0%	443,4	0,4%	444,6	0,7%	446,6	1,1%
456,4	458,4	0,4%	457,6	0,3%	459,4	0,7%	456,4	0,0%
406,6	407,2	0,1%	406,6	0,0%	412,4	1,4%	410	0,8%
309,6	314,8	1,7%	310	0,1%	313,4	1,2%	309,6	0,0%
428,8	435,2	1,5%	436,4	1,8%	428,8	0,0%	431,6	0,7%
413,8	417,8	1,0%	419,4	1,4%	413,8	0,0%	416,2	0,6%
306,8	306,8	0,0%	306,8	0,0%	308,2	0,5%	307	0,1%
471	471	0,0%	472,8	0,4%	472,4	0,3%	474	0,6%
442	445	0,7%	444,8	0,6%	442	0,0%	445	0,7%
376,8	380,6	1,0%	376,8	0,0%	380,6	1,0%	376,8	0,0%
405,6	405,6	0,0%	408,6	0,7%	412,6	1,7%	410	1,1%
418,2	420,8	0,6%	422	0,9%	421	0,7%	418,2	0,0%
396,2	396,2	0,0%	410,2	3,5%	406,2	2,5%	397,2	0,3%
281,6	286,4	1,7%	285,6	1,4%	281,6	0,0%	291,6	3,6%
434,4	437,6	0,7%	436,4	0,5%	437,2	0,6%	434,4	0,0%
390,2	391,8	0,4%	394,2	1,0%	390,2	0,0%	391,8	0,4%
296	297	0,3%	296	0,0%	296,6	0,2%	296,6	0,2%
444,8	444,8	0,0%	450	1,2%	451,4	1,5%	449	0,9%
418,2	418,2	0,0%	421,6	0,8%	419,6	0,3%	423,6	1,3%
382,6	386,6	1,0%	389,8	1,9%	385,8	0,8%	382,6	0,0%
389	389	0,0%	389,4	0,1%	390,8	0,5%	398,6	2,5%
377,2	382	1,3%	385,6	2,2%	380,6	0,9%	377,2	0,0%
598,4	598,4	0,0%	601,4	0,5%	613,6	2,5%	602,4	0,7%
493,8	505	2,3%	509	3,1%	501	1,5%	493,8	0,0%
484	490,8	1,4%	486	0,4%	490,2	1,3%	484	0,0%
502	515,2	2,6%	512,2	2,0%	510,8	1,8%	502	0,0%
475,4	475,4	0,0%	479,6	0,9%	475,4	0,0%	484,6	1,9%
499,4	507,4	1,6%	504	0,9%	499,4	0,0%	501,2	0,4%
432,4	442,8	2,4%	441,8	2,2%	446,8	3,3%	432,4	0,0%
552	552	0,0%	560,2	1,5%	559,2	1,3%	552,2	0,0%
540	540	0,0%	546,4	1,2%	548,2	1,5%	541,4	0,3%
323	324,8	0,6%	329,4	2,0%	326,6	1,1%	323	0,0%
558,6	558,6	0,0%	566,6	1,4%	564	1,0%	562,6	0,7%
478	491,2	2,8%	485	1,5%	486,4	1,8%	478	0,0%
447,2	455,8	1,9%	459	2,6%	455,2	1,8%	447,2	0,0%
474,4	474,4	0,0%	477,8	0,7%	481,8	1,6%	478,8	0,9%
444,8	455,4	2,4%	459	3,2%	444,8	0,0%	449,6	1,1%
448,2	448,2	0,0%	458	2,2%	454,8	1,5%	450,6	0,5%
413	414,6	0,4%	413	0,0%	414	0,2%	418,4	1,3%
513,6	521,2	1,5%	528,2	2,8%	513,6	0,0%	530	3,2%
504,2	504,2	0,0%	514,6	2,1%	507,2	0,6%	510,2	1,2%

Jobs	Ie	Var	Time	Ie	Var	Time	Ie	Var	Time	Ie	Var	Time
20	2,24%	0,00074	9,8	2,61%	0,00084	10,4	2,05%	0,00063	18,5	2,29%	0,00059	17,7
50	1,88%	0,00046	33,0	2,01%	0,00062	33,2	1,81%	0,00047	58,6	1,54%	0,00035	55,9

RSAI

MIN	Ex1	Ie	Ex2	Ie	Ex3	Ie	Ex4	Ie
348,8	350,2	0,4%	350	0,3%	348,8	0,0%	354,4	1,6%
445,8	448,4	0,6%	451,8	1,3%	451,2	1,2%	445,8	0,0%
378,2	378,2	0,0%	386,2	2,1%	389,6	3,0%	394,4	4,3%
405,6	427,2	5,3%	419	3,3%	405,6	0,0%	408,4	0,7%
454,2	487	7,2%	463,2	2,0%	471,2	3,7%	454,2	0,0%
331	335,2	1,3%	339,8	2,7%	331	0,0%	335,4	1,3%
486,4	498,8	2,5%	498,8	2,5%	486,4	0,0%	489,2	0,6%
375	380,8	1,5%	391,4	4,4%	386,8	3,1%	375	0,0%
437,4	453,8	3,7%	440,2	0,6%	440,4	0,7%	437,4	0,0%
463,8	463,8	0,0%	466,4	0,6%	478,4	3,1%	487,6	5,1%

MIN	Ex1	Ie	Ex2	Ie	Ex3	Ie	Ex4	Ie
326,6	326,6	0,0%	330,8	1,3%	338,6	3,7%	328,8	0,7%
406,4	406,4	0,0%	407,6	0,3%	416,4	2,5%	428,4	5,4%
330,4	342,2	3,6%	330,4	0,0%	350	5,9%	351,6	6,4%
369	369	0,0%	370,2	0,3%	386,2	4,7%	380,8	3,2%
426,6	449	5,3%	436	2,2%	427,8	0,3%	426,6	0,0%
300,6	304	1,1%	310,2	3,2%	304,4	1,3%	300,6	0,0%
436,2	457,8	5,0%	436,2	0,0%	443,6	1,7%	438,4	0,5%
365,8	373	2,0%	370,8	1,4%	365,8	0,0%	373,8	2,2%
434,8	434,8	0,0%	437,6	0,6%	438	0,7%	435,8	0,2%
455,4	465,4	2,2%	460	1,0%	455,4	0,0%	463,4	1,8%
323,4	331,2	2,4%	327,6	1,3%	323,4	0,0%	331	2,4%
398	400,2	0,6%	401	0,8%	400,4	0,6%	398	0,0%
313,6	313,6	0,0%	318	1,4%	333,4	6,3%	336,2	7,2%
371,6	398	7,1%	376,4	1,3%	371,6	0,0%	376,4	1,3%
418	432,2	3,4%	422,4	1,1%	418	0,0%	423,4	1,3%
290,6	294,8	1,4%	290,6	0,0%	292,6	0,7%	291,8	0,4%
393,4	393,4	0,0%	418,8	6,5%	417	6,0%	408	3,7%
376,8	376,8	0,0%	386,8	2,7%	376,8	0,0%	394	4,6%
417,2	417,2	0,0%	429,2	2,9%	443,4	6,3%	425,8	2,1%
447,8	447,8	0,0%	455,6	1,7%	470,4	5,0%	457,2	2,1%
329,4	329,4	0,0%	339,2	3,0%	330,8	0,4%	333	1,1%
410,8	421,8	2,7%	420,2	2,3%	416,4	1,4%	410,8	0,0%
315,4	317,6	0,7%	315,6	0,1%	321,2	1,8%	315,4	0,0%
366,6	366,6	0,0%	374,8	2,2%	378,4	3,2%	379,2	3,4%
437,2	447,2	2,3%	437,2	0,0%	450,2	3,0%	441,6	1,0%
285,6	315,2	10,4%	285,6	0,0%	313	9,6%	314,4	10,1%
406	406,6	0,1%	416,8	2,7%	406	0,0%	407,2	0,3%
395	395	0,0%	410,6	3,9%	402,6	1,9%	422	6,8%
431,6	431,6	0,0%	437,2	1,3%	441	2,2%	453,2	5,0%
456,8	456,8	0,0%	458,4	0,4%	468,8	2,6%	459,8	0,7%
435,4	435,4	0,0%	439,6	1,0%	447,2	2,7%	440,6	1,2%
374,4	374,4	0,0%	386,2	3,2%	376	0,4%	381	1,8%
451,6	452,2	0,1%	451,6	0,0%	452,6	0,2%	462,2	2,3%
469,2	473,4	0,9%	474,2	1,1%	470,6	0,3%	469,2	0,0%
359,2	359,2	0,0%	370,8	3,2%	372	3,6%	375,4	4,5%
541,2	541,2	0,0%	541,8	0,1%	547,8	1,2%	559,2	3,3%
506,2	513	1,3%	508	0,4%	506,2	0,0%	521,4	3,0%
415	415	0,0%	422,8	1,9%	423,2	2,0%	423	1,9%
483,8	486,4	0,5%	495,4	2,4%	483,8	0,0%	494,8	2,3%
499	499	0,0%	506	1,4%	508	1,8%	508,8	2,0%
426,4	437	2,5%	436,2	2,3%	426,4	0,0%	436,2	2,3%
346	347	0,3%	349	0,9%	360,6	4,2%	346	0,0%
442,6	442,6	0,0%	447,4	1,1%	450,2	1,7%	464,6	5,0%
441,8	453,4	2,6%	441,8	0,0%	452,2	2,4%	445,2	0,8%
340,6	346,8	1,8%	344,8	1,2%	340,6	0,0%	342,6	0,6%
485,8	514,2	5,8%	522,4	7,5%	510,6	5,1%	485,8	0,0%
471,2	476,4	1,1%	481	2,1%	482	2,3%	471,2	0,0%
392,4	410,6	4,6%	395,6	0,8%	404	3,0%	392,4	0,0%
450,4	454,4	0,9%	450,4	0,0%	452,4	0,4%	453,4	0,7%
466,8	474,8	1,7%	466,8	0,0%	467,6	0,2%	478,2	2,4%
417,4	417,4	0,0%	420,4	0,7%	421,8	1,1%	431,4	3,4%
312,8	319	2,0%	322,4	3,1%	314,4	0,5%	312,8	0,0%
436,6	449,8	3,0%	436,6	0,0%	446,4	2,2%	441,6	1,1%
418,4	424	1,3%	418,4	0,0%	422,8	1,1%	422,6	1,0%
307	307	0,0%	327,4	6,6%	312,6	1,8%	330,8	7,8%
472,8	482	1,9%	472,8	0,0%	482	1,9%	478,2	1,1%
446,6	451,8	1,2%	465,2	4,2%	446,6	0,0%	450	0,8%
356,6	382	7,1%	356,6	0,0%	394	10,5%	379,8	6,5%
414,6	419,6	1,2%	414,6	0,0%	416,4	0,4%	418,2	0,9%
421,4	435,4	3,3%	421,4	0,0%	434	3,0%	429	1,8%
411,4	417,6	1,5%	428,4	4,1%	411,4	0,0%	415,8	1,1%
287,6	297,2	3,3%	296,8	3,2%	287,6	0,0%	297,8	3,5%
438,4	442,2	0,9%	442,4	0,9%	451,6	3,0%	438,4	0,0%
397	397,8	0,2%	413,8	4,2%	400	0,8%	397	0,0%
299,8	328,2	9,5%	307	2,4%	306,2	2,1%	299,8	0,0%
453,2	453,2	0,0%	463,4	2,3%	470,6	3,8%	463,2	2,2%
428,6	437,8	2,1%	448,6	4,7%	445	3,8%	428,6	0,0%

MIN	Ex1	Ie	Ex2	Ie	Ex3	Ie	Ex4	Ie
388,4	388,4	0,0%	405,4	4,4%	388,6	0,1%	391,6	0,8%
398,4	412,4	3,5%	398,4	0,0%	410,2	3,0%	414,4	4,0%
391	391	0,0%	402,8	3,0%	395,8	1,2%	404	3,3%
605,6	623	2,9%	621,8	2,7%	605,6	0,0%	627,6	3,6%
516	516	0,0%	525	1,7%	523,2	1,4%	526,2	2,0%
498,2	498,2	0,0%	508,8	2,1%	499,4	0,2%	511,2	2,6%
528,2	542,8	2,8%	542,2	2,7%	528,8	0,1%	528,2	0,0%
482,8	499,6	3,5%	491,4	1,8%	483,6	0,2%	482,8	0,0%
507,4	515,4	1,6%	507,4	0,0%	518	2,1%	528,4	4,1%
438,8	451,8	3,0%	463,8	5,7%	453,2	3,3%	438,8	0,0%
546,8	569,2	4,1%	546,8	0,0%	572,2	4,6%	576,8	5,5%
545,8	555,2	1,7%	552,2	1,2%	558,6	2,3%	545,8	0,0%
323,4	327,4	1,2%	323,4	0,0%	330,2	2,1%	337,8	4,5%
524,6	524,6	0,0%	541	3,1%	591	12,7%	585,6	11,6%
445,4	509	14,3%	527,2	18,4%	445,4	0,0%	488,6	9,7%
463,2	463,2	0,0%	479,6	3,5%	467,8	1,0%	464,6	0,3%
480,8	480,8	0,0%	500,2	4,0%	483,8	0,6%	510,6	6,2%
448	448	0,0%	463,6	3,5%	455,2	1,6%	463,8	3,5%
465,6	469,8	0,9%	480,2	3,1%	465,6	0,0%	477,8	2,6%
419,8	421,2	0,3%	421	0,3%	419,8	0,0%	437	4,1%
537,2	540	0,5%	540	0,5%	537,2	0,0%	545	1,5%
519,2	519,4	0,0%	531,4	2,3%	520,6	0,3%	519,2	0,0%

Phase 2: Performance Experiments

For the complete results, see the archive Phase 2 Performance.xls on the CD.

First, the output data for 30 instances for each configuration of jobs and machines is given for EDD SAS. Second, the summary as well as an example of the calculations is given for each of the 6 versions of the algorithm. The example is presented as the result from 25 instances from each of the 5 configurations of jobs and machines.

EDD SAS

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	Cmaxbest	Tmaxbest	Time	it
15	2	1	100	1,2	0,8	0,2	10	491,6	0	395,4	528	307	13,125	538025
15	2	2	100	1,2	0,8	0,2	10	422	0	339,6	465	256	1,71875	70276
15	2	3	100	1,2	0,8	0,2	10	416,8	0	309,4	451	215	3,375	138588
15	2	4	100	1,2	0,8	0,2	10	583,4	0	488,6	626	397	1,375	55816
15	2	5	100	1,2	0,8	0,2	10	509,8	0	378	507	292	3,859375	159215
15	2	6	100	1,2	0,8	0,2	10	503	0	392,8	529	302	7,953125	325467
15	2	7	100	1,2	0,8	0,2	10	456,4	0	354,4	478	272	1,953125	79641
15	2	8	100	1,2	0,8	0,2	10	477,4	0	393,8	533	301	2,46875	101720
15	2	9	100	1,2	0,8	0,2	10	481,8	0	405,4	517	331	12,65625	518394
15	2	10	100	1,2	0,8	0,2	10	481	0	349,8	486	259	3,421875	139688
15	2	1	100	1,2	0,8	0,2	10	491,6	0	396,2	530	307	1,90625	77653
15	2	2	100	1,2	0,8	0,2	10	422	0	341,2	463	260	2,15625	88530
15	2	3	100	1,2	0,8	0,2	10	416,8	0	310,6	451	217	2,625	107612
15	2	4	100	1,2	0,8	0,2	10	583,4	0	484,2	627	389	1,03125	42097
15	2	5	100	1,2	0,8	0,2	10	509,8	0	383	518	293	1,546875	63305
15	2	6	100	1,2	0,8	0,2	10	503	0	392	527	302	3,71875	152745
15	2	7	100	1,2	0,8	0,2	10	456,4	0	355,8	480	273	7,75	316850
15	2	8	100	1,2	0,8	0,2	10	477,4	0	390,4	532	296	1,8125	74406
15	2	9	100	1,2	0,8	0,2	10	481,8	0	405,6	525	326	3,65625	149906
15	2	10	100	1,2	0,8	0,2	10	481	0	354,6	498	259	5,21875	213533
15	2	1	100	1,2	0,8	0,2	10	491,6	0	396,2	530	307	1,5	61114
15	2	2	100	1,2	0,8	0,2	10	422	0	340	472	252	2,890625	118888
15	2	3	100	1,2	0,8	0,2	10	416,8	0	319,8	462	225	2,015625	82822
15	2	4	100	1,2	0,8	0,2	10	583,4	0	483,8	626	389	13,875	568902
15	2	5	100	1,2	0,8	0,2	10	509,8	0	383,2	517	294	2,625	107497
15	2	6	100	1,2	0,8	0,2	10	503	0	386	524	294	1	40643
15	2	7	100	1,2	0,8	0,2	10	456,4	0	356,4	480	274	12,71875	520838
15	2	8	100	1,2	0,8	0,2	10	477,4	0	399,2	536	308	1,796875	73877
15	2	9	100	1,2	0,8	0,2	10	481,8	0	401,6	521	322	8,3125	340665
15	2	10	100	1,2	0,8	0,2	10	481	0	352	487	262	2,28125	93684
20	2	1	100	1,2	0,8	0,2	10	420,4	0	342,4	499	238	2,234375	70904
20	2	2	100	1,2	0,8	0,2	10	532	0	438,8	620	318	4,640625	147771
20	2	3	100	1,2	0,8	0,2	10	474,6	0	372,4	556	250	1,453125	46169
20	2	4	100	1,2	0,8	0,2	10	477	0	406	586	286	7,859375	250017
20	2	5	100	1,2	0,8	0,2	10	581,2	0	462,2	644	341	1,21875	38878
20	2	6	100	1,2	0,8	0,2	10	404,2	0	327,2	509	206	5,421875	171983
20	2	7	100	1,2	0,8	0,2	10	579,6	0	479,2	658	360	8,625	275616
20	2	8	100	1,2	0,8	0,2	10	442,6	0	375,2	554	256	2,234375	71332
20	2	9	100	1,2	0,8	0,2	10	544,8	0	430,2	615	307	3,0625	97414
20	2	10	100	1,2	0,8	0,2	10	538,2	0	457	640	335	3,78125	120175
20	2	1	100	1,2	0,8	0,2	10	420,4	0	337,6	499	230	5,078125	162155
20	2	2	100	1,2	0,8	0,2	10	532	0	449	629	329	4,8125	153370
20	2	3	100	1,2	0,8	0,2	10	474,6	0	374,6	551	257	4,1875	133028
20	2	4	100	1,2	0,8	0,2	10	477	0	392,6	575	271	2,4375	77973
20	2	5	100	1,2	0,8	0,2	10	581,2	0	452,8	637	330	5,1875	164586
20	2	6	100	1,2	0,8	0,2	10	404,2	0	337,6	517	218	24,46875	776084
20	2	7	100	1,2	0,8	0,2	10	579,6	0	477,2	659	356	3	95710
20	2	8	100	1,2	0,8	0,2	10	442,6	0	385,2	564	266	18,90625	602059
20	2	9	100	1,2	0,8	0,2	10	544,8	0	429,4	616	305	2,296875	72842
20	2	10	100	1,2	0,8	0,2	10	538,2	0	460,4	635	344	1,8125	57966
20	2	1	100	1,2	0,8	0,2	10	420,4	0	351,8	509	247	7,125	227428

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	Cmaxbest	Tmaxbest	Time	it
20	2	2	100	1,2	0,8	0,2	10	532	0	445	625	325	3,28125	104478
20	2	3	100	1,2	0,8	0,2	10	474,6	0	379,2	558	260	5,640625	179675
20	2	4	100	1,2	0,8	0,2	10	477	0	404,6	587	283	3,03125	96668
20	2	5	100	1,2	0,8	0,2	10	581,2	0	459,8	638	341	1,96875	62407
20	2	6	100	1,2	0,8	0,2	10	404,2	0	329,4	504	213	2,53125	80383
20	2	7	100	1,2	0,8	0,2	10	579,6	0	484,8	663	366	1,03125	32930
20	2	8	100	1,2	0,8	0,2	10	442,6	0	370,4	548	252	1,71875	54481
20	2	9	100	1,2	0,8	0,2	10	544,8	0	437,8	625	313	7	222787
20	2	10	100	1,2	0,8	0,2	10	538,2	0	461	638	343	17,875	568874
20	3	1	100	1,2	0,8	0,2	10	457,8	0	362,6	479	285	2,90625	64833
20	3	2	100	1,2	0,8	0,2	10	317,4	0	238,6	352	163	3,546875	79218
20	3	3	100	1,2	0,8	0,2	10	413,4	0	370,2	492	289	6,5	145591
20	3	4	100	1,2	0,8	0,2	10	423,8	0	312	438	228	4,484375	100457
20	3	5	100	1,2	0,8	0,2	10	346,2	0	271,2	393	190	2,828125	62994
20	3	6	100	1,2	0,8	0,2	10	383,8	0	314,4	426	240	2,625	58915
20	3	7	100	1,2	0,8	0,2	10	397,4	0	314,4	417	246	6,265625	140348
20	3	8	100	1,2	0,8	0,2	10	337,2	0	265,6	385	186	3,140625	69903
20	3	9	100	1,2	0,8	0,2	10	384,2	0	303,4	409	233	13,70313	307538
20	3	10	100	1,2	0,8	0,2	10	289,6	0	242,2	361	163	5,515625	123358
20	3	1	100	1,2	0,8	0,2	10	457,8	0	367,6	478	294	2,6875	60074
20	3	2	100	1,2	0,8	0,2	10	317,4	0	238,2	351	163	23,6875	528257
20	3	3	100	1,2	0,8	0,2	10	413,4	0	372,2	494	291	5,5625	124362
20	3	4	100	1,2	0,8	0,2	10	423,8	0	319,6	442	238	1,78125	40032
20	3	5	100	1,2	0,8	0,2	10	346,2	0	274,4	392	196	15,73438	351045
20	3	6	100	1,2	0,8	0,2	10	383,8	0	320,4	429	248	4,25	95170
20	3	7	100	1,2	0,8	0,2	10	397,4	0	319,6	427	248	4,734375	105690
20	3	8	100	1,2	0,8	0,2	10	337,2	0	271,2	387	194	19,60938	437972
20	3	9	100	1,2	0,8	0,2	10	384,2	0	309,2	431	228	3	66993
20	3	10	100	1,2	0,8	0,2	10	289,6	0	245,6	365	166	9,09375	203434
20	3	1	100	1,2	0,8	0,2	10	457,8	0	363,8	476	289	5,921875	132626
20	3	2	100	1,2	0,8	0,2	10	317,4	0	233,6	341	162	13,73438	306024
20	3	3	100	1,2	0,8	0,2	10	413,4	0	371,8	493	291	6,515625	145874
20	3	4	100	1,2	0,8	0,2	10	423,8	0	319,8	441	239	5,75	128610
20	3	5	100	1,2	0,8	0,2	10	346,2	0	271,2	378	200	5,390625	120705
20	3	6	100	1,2	0,8	0,2	10	383,8	0	322,4	440	244	16,96875	379702
20	3	7	100	1,2	0,8	0,2	10	397,4	0	312	426	236	11,23438	251127
20	3	8	100	1,2	0,8	0,2	10	337,2	0	271,2	393	190	21,46875	478754
20	3	9	100	1,2	0,8	0,2	10	384,2	0	308,6	425	231	7,546875	168813
20	3	10	100	1,2	0,8	0,2	10	289,6	0	240,6	360	161	6,28125	140829
50	3	1	100	1,2	0,8	0,2	10	804	0	674	995	460	20,92188	196675
50	3	2	100	1,2	0,8	0,2	10	840,6	0	724	1042	512	11,71875	110193
50	3	3	100	1,2	0,8	0,2	10	878,2	0	766	1087	552	11,74219	110361
50	3	4	100	1,2	0,8	0,2	10	802,6	0	702,2	1025	487	17,98438	168700
50	3	5	100	1,2	0,8	0,2	10	765,6	0	677,4	999	463	16,75781	157504
50	3	6	100	1,2	0,8	0,2	10	767,2	0	662,8	982	450	12,34375	115960
50	3	7	100	1,2	0,8	0,2	10	710,6	0	599,2	922	384	17,09375	161055
50	3	8	100	1,2	0,8	0,2	10	764	0	637	946	431	17,04688	160514
50	3	9	100	1,2	0,8	0,2	10	807,2	0	692,8	1015	478	15,73438	148020
50	3	10	100	1,2	0,8	0,2	10	713,2	0	592,6	910	381	14,92188	140552
50	3	1	100	1,2	0,8	0,2	10	804	0	676,8	999	462	14,32031	134614
50	3	2	100	1,2	0,8	0,2	10	840,6	0	714,6	1032	503	23,65625	222562
50	3	3	100	1,2	0,8	0,2	10	878,2	0	753,2	1079	536	20	188246
50	3	4	100	1,2	0,8	0,2	10	802,6	0	708,6	1029	495	9,640625	90422
50	3	5	100	1,2	0,8	0,2	10	765,6	0	681,4	1003	467	15,09375	142042
50	3	6	100	1,2	0,8	0,2	10	767,2	0	662,2	985	447	14,5	136356
50	3	7	100	1,2	0,8	0,2	10	710,6	0	596,6	920	381	13,46094	126705
50	3	8	100	1,2	0,8	0,2	10	764	0	653,4	957	451	8,15625	76763
50	3	9	100	1,2	0,8	0,2	10	807,2	0	689,8	1012	475	24,9375	234518
50	3	10	100	1,2	0,8	0,2	10	713,2	0	597,8	911	389	16,90625	159262
50	3	1	100	1,2	0,8	0,2	10	804	0	672,2	995	457	15,53125	145932
50	3	2	100	1,2	0,8	0,2	10	840,6	0	721,2	1041	508	10,10938	95089
50	3	3	100	1,2	0,8	0,2	10	878,2	0	761,6	1082	548	15,42969	145086
50	3	4	100	1,2	0,8	0,2	10	802,6	0	697,2	1023	480	20,51563	192605
50	3	5	100	1,2	0,8	0,2	10	765,6	0	682,8	1005	468	8,234375	77551
50	3	6	100	1,2	0,8	0,2	10	767,2	0	657,8	980	443	13,45313	126505
50	3	7	100	1,2	0,8	0,2	10	710,6	0	607	931	391	16,15625	152204
50	3	8	100	1,2	0,8	0,2	10	764	0	646,4	953	442	14,04688	132326

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	Cmaxbest	Tmaxbest	Time	it
50	3	9	100	1,2	0,8	0,2	10	807,2	0	682,2	1005	467	11,71875	110146
50	3	10	100	1,2	0,8	0,2	10	713,2	0	601,4	917	391	13,71875	129116
50	4	1	100	1,2	0,8	0,2	10	567,8	0	493,4	734	333	15,75	112702
50	4	2	100	1,2	0,8	0,2	10	574,4	0	516,6	756	357	24,73438	176859
50	4	3	100	1,2	0,8	0,2	10	627	0	561,8	803	401	27,875	199410
50	4	4	100	1,2	0,8	0,2	10	625,4	0	553,4	794	393	27,6875	197982
50	4	5	100	1,2	0,8	0,2	10	616,8	0	525,2	764	366	19,90625	142525
50	4	6	100	1,2	0,8	0,2	10	621,8	0	507,8	746	349	15,92188	113938
50	4	7	100	1,2	0,8	0,2	10	576,8	0	503,8	748	341	20,53125	146770
50	4	8	100	1,2	0,8	0,2	10	421,8	0	361,2	588	210	12,09375	86479
50	4	9	100	1,2	0,8	0,2	10	512,4	0	445,4	680	289	14	100220
50	4	10	100	1,2	0,8	0,2	10	531,6	0	455,8	691	299	23,35938	167082
50	4	1	100	1,2	0,8	0,2	10	567,8	0	490,4	731	330	22,125	158361
50	4	2	100	1,2	0,8	0,2	10	574,4	0	517,8	762	355	24,89063	177959
50	4	3	100	1,2	0,8	0,2	10	627	0	561,6	804	400	17,1875	122899
50	4	4	100	1,2	0,8	0,2	10	625,4	0	550,8	792	390	28,6875	205248
50	4	5	100	1,2	0,8	0,2	10	616,8	0	513,6	753	354	29,85938	213817
50	4	6	100	1,2	0,8	0,2	10	621,8	0	505,2	741	348	19,73438	141210
50	4	7	100	1,2	0,8	0,2	10	576,8	0	500,8	742	340	21,25	151912
50	4	8	100	1,2	0,8	0,2	10	421,8	0	363,6	594	210	19,78125	141484
50	4	9	100	1,2	0,8	0,2	10	512,4	0	442	676	286	17,46875	125043
50	4	10	100	1,2	0,8	0,2	10	531,6	0	452,6	689	295	27,84375	199222
50	4	1	100	1,2	0,8	0,2	10	567,8	0	491	731	331	11,51563	82408
50	4	2	100	1,2	0,8	0,2	10	574,4	0	519,2	761	358	29,51563	211022
50	4	3	100	1,2	0,8	0,2	10	627	0	561	798	403	22,21875	158831
50	4	4	100	1,2	0,8	0,2	10	625,4	0	546,4	787	386	31,51563	225320
50	4	5	100	1,2	0,8	0,2	10	616,8	0	526,2	765	367	17,57813	125901
50	4	6	100	1,2	0,8	0,2	10	621,8	0	504,4	742	346	13,35938	95630
50	4	7	100	1,2	0,8	0,2	10	576,8	0	502,8	747	340	25,25	180557
50	4	8	100	1,2	0,8	0,2	10	421,8	0	362	590	210	18,875	135087
50	4	9	100	1,2	0,8	0,2	10	512,4	0	442,4	674	288	23,96875	171575
50	4	10	100	1,2	0,8	0,2	10	531,6	0	466,2	693	315	23,90625	171131

Ie	1,99%	Iterations	204 779	Ie	1,93%	Iterations	202 153	Ie	5,15%	Iterations	249 853
Variance	0,00032	Time (s)	14,0	Variance	0,00030	Time (s)	13,9	Variance	0,00049	Time (s)	18,3
15,2	2,60%		209 222	15,2	2,51%		205 144	15,2	5,07%		235 205
	0,00081		5,1		0,00081		5,0		0,00080		6,2
20,2	2,27%		211 165	20,2	2,21%		206 004	20,2	4,68%		242 705
	0,00033		6,6		0,00031		6,5		0,00041		8,2
20,3	2,21%		206 530	20,3	2,01%		201 469	20,3	5,82%		253 173
	0,00027		9,2		0,00021		9,0		0,00040		11,7
50,3	1,46%		188 663	50,3	1,51%		193 928	50,3	4,49%		246 707
	0,00005		20,1		0,00005		20,6		0,00029		27,0
50,4	1,40%		208 317	50,4	1,41%		204 220	50,4	5,70%		271 475
	0,00004		29,1		0,00003		28,6		0,00041		38,2

			ESAS			**RSAS**			**ESAI**		
n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie	Min	Avarage	Ie
15	2	393,6	395,4	396,56	0,8%	394,6	399,08	1,4%	402,4	407,88	3,6%
15	2	331,4	338	340,84	2,8%	335,2	337,2	1,8%	334,4	338,28	2,1%
15	2	302,8	309,4	312,68	3,3%	302,8	309,72	2,3%	311,6	321,8	6,3%
15	2	439,4	483	485,36	10,5%	476	479,84	9,2%	500,6	506,24	15,2%
15	2	348,6	378	382,52	9,7%	383,2	384,56	10,3%	383,2	394,92	13,3%
15	2	384,4	386	391,28	1,8%	384,4	390,48	1,6%	395,2	400,28	4,1%
15	2	342,8	349,2	353,04	3,0%	344,2	348	1,5%	347,4	354,6	3,4%
15	2	390,4	390,4	395,72	1,4%	393,2	397,52	1,8%	398,6	404,68	3,7%
15	2	396,6	396,6	401,2	1,2%	396,6	398	0,4%	399,2	410,76	3,6%
15	2	337	349,8	353,16	4,8%	345,4	350,84	4,1%	344,2	353,4	4,9%
15	2	370	375,6	377,36	2,0%	370	375,48	1,5%	380	389,52	5,3%
15	2	285,4	316,8	320,32	12,2%	318,8	320,76	12,4%	285,4	322,52	13,0%
15	2	290,6	290,6	297,52	2,4%	293,6	296,36	2,0%	294,4	299,96	3,2%
15	2	440,6	441,4	444,64	0,9%	441,4	445,44	1,1%	457,4	466	5,8%
15	2	346,8	371,8	372,2	7,3%	371,8	379,52	9,4%	346,8	379,96	9,6%
15	2	350,8	351,8	358	2,1%	354,8	359,96	2,6%	361	370,72	5,7%
15	2	320,4	320,4	325,92	1,7%	324,2	327,64	2,3%	325,4	331,08	3,3%
15	2	357,4	357,4	361,56	1,2%	357,4	362,32	1,4%	362,4	371,44	3,9%
15	2	391,6	394,2	394,88	0,8%	391,6	393,96	0,6%	398,2	404,72	3,4%

n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie	Min	Avarage	Ie
15	2	312	312,6	314,04	0,7%	312	312,96	0,3%	317	321,56	3,1%
15	2	347,2	347,2	348,52	0,4%	349,2	351,36	1,2%	355,2	362,4	4,4%
15	2	290,2	307,6	312,8	7,8%	308,4	310,68	7,1%	306,8	316,64	9,1%
15	2	300,2	300,2	303,48	1,1%	300,2	304,44	1,4%	300,2	309,8	3,2%
15	2	400,4	400,4	402,08	0,4%	400,4	402,96	0,6%	401,6	416,4	4,0%
15	2	382,4	382,4	384,16	0,5%	382,4	383,64	0,3%	386,2	392,64	2,7%
20	2	337,6	337,6	346,68	2,7%	343	348,72	3,3%	350,8	353,6	4,7%
20	2	434	438,8	444,6	2,4%	438	442,56	2,0%	446,4	450,56	3,8%
20	2	372,2	372,4	375,2	0,8%	372,2	376,84	1,2%	379,2	383,48	3,0%
20	2	392,6	392,6	402,16	2,4%	394	404,04	2,9%	395	407,36	3,8%
20	2	451,2	452,8	457,08	1,3%	451,2	459,88	1,9%	463,8	473,32	4,9%
20	2	323,6	327,2	331,6	2,5%	329,4	332,84	2,9%	326,4	336,76	4,1%
20	2	472,8	473,2	478,44	1,2%	473,8	481,08	1,8%	483,8	490,32	3,7%
20	2	367,2	369	375,56	2,3%	370,2	378,64	3,1%	374	382,88	4,3%
20	2	423,8	429,4	431,8	1,9%	423,8	433,56	2,3%	430,4	439,72	3,8%
20	2	440,2	457	460,08	4,5%	451,2	458,36	4,1%	458,6	467,68	6,2%
20	2	291,4	322,6	326,88	12,2%	326,2	328,96	12,9%	315,2	329,12	12,9%
20	2	399,2	401	403,64	1,1%	399,2	403,4	1,1%	410	419,04	5,0%
20	2	328	337,6	341,2	4,0%	333	336,28	2,5%	346,2	354,24	8,0%
20	2	356	359,8	371,32	4,3%	361,8	365,96	2,8%	359,2	372,88	4,7%
20	2	398,4	410	414,12	3,9%	417,4	419,68	5,3%	398,4	419,96	5,4%
20	2	297,2	300	302,04	1,6%	300	304,12	2,3%	301,4	308,44	3,8%
20	2	425	427	431,4	1,5%	425	430,44	1,3%	433,8	444,04	4,5%
20	2	359,4	359,4	365,88	1,8%	364,2	365,72	1,8%	364,2	370,84	3,2%
20	2	414,2	418	420,52	1,5%	421	422,36	2,0%	414,2	430,36	3,9%
20	2	450,4	457	459,84	2,1%	454,6	455,44	1,1%	454,6	462	2,6%
20	2	316,4	320,2	323,72	2,3%	320	323,4	2,2%	317,4	325,28	2,8%
20	2	393,6	393,6	397,84	1,1%	399,6	402,8	2,3%	407,2	416,12	5,7%
20	2	308	312	313,8	1,9%	310,2	313,64	1,8%	312,6	327,28	6,3%
20	2	365,8	365,8	368,52	0,7%	368,8	371,76	1,6%	371,2	373,08	2,0%
20	2	409,4	409,4	412,12	0,7%	412,6	414,92	1,3%	423,2	427,76	4,5%
20	3	357,6	357,6	363,64	1,7%	361	363,4	1,6%	369,6	375,44	5,0%
20	3	225,8	233,6	235,92	4,5%	232,4	235,08	4,1%	231,6	241,8	7,1%
20	3	364,4	369	371,08	1,8%	364,4	370,08	1,6%	373,2	383	5,1%
20	3	309,8	312	317,16	2,4%	316,2	321,72	3,8%	322,6	327,96	5,9%
20	3	263,4	263,4	269,08	2,2%	267,2	270,32	2,6%	274,6	278,28	5,6%
20	3	312,8	314,4	319,88	2,3%	312,8	318,44	1,8%	328,8	333,6	6,6%
20	3	305,8	312	314,48	2,8%	307,6	311,4	1,8%	312,4	319,88	4,6%
20	3	265,6	265,6	268,84	1,2%	268,2	269,72	1,6%	278,6	282,24	6,3%
20	3	287,2	303,4	307,16	6,9%	302,8	305,84	6,5%	287,2	305,32	6,3%
20	3	234,8	240,6	242,72	3,4%	236	240,08	2,2%	245,8	251,76	7,2%
20	3	344,8	346	351,2	1,9%	346,8	351	1,8%	356,8	359,96	4,4%
20	3	225	228,6	233,68	3,9%	232,4	234,6	4,3%	234,8	241,24	7,2%
20	3	330,8	333,6	339,28	2,6%	337,6	342,48	3,5%	330,8	352,88	6,7%
20	3	282,2	284,8	287,2	1,8%	282,2	286,6	1,6%	288,4	294,36	4,3%
20	3	240,6	246	248,44	3,3%	245,6	248,4	3,2%	249,4	255,68	6,3%
20	3	297	301,6	303,52	2,2%	299,6	303,32	2,1%	304,6	314	5,7%
20	3	292	297	298,84	2,3%	293,2	299,04	2,4%	300,2	306,8	5,1%
20	3	246,2	247	249,96	1,5%	248,4	250,32	1,7%	256,4	261,32	6,1%
20	3	281,6	289	291,04	3,4%	289	291,8	3,6%	288	298,08	5,9%
20	3	218,2	222,6	225,44	3,3%	223	224,96	3,1%	229,8	234,64	7,5%
20	3	337,2	341,6	343,6	1,9%	337,2	339,36	0,6%	353	359,44	6,6%
20	3	225,6	231,2	240,08	6,4%	228	234,28	3,8%	229	247,2	9,6%
20	3	316,2	317,8	319,48	1,0%	316,2	325,28	2,9%	334,4	338,48	7,0%
20	3	251,2	251,4	256,8	2,2%	255,2	258	2,7%	266,2	273,08	8,7%
20	3	226,2	231,6	234,12	3,5%	231	235,08	3,9%	235,6	245,16	8,4%
50	3	663	663,6	670,76	1,2%	663	670,68	1,2%	669,4	687,76	3,7%
50	3	704,8	706,4	715,96	1,6%	704,8	716,48	1,7%	719,2	738,48	4,8%
50	3	751	753,2	760,92	1,3%	751	761,6	1,4%	770,6	785,12	4,5%
50	3	686,6	697,2	703,68	2,5%	695	703,56	2,5%	706,2	713,6	3,9%
50	3	667	667	678,24	1,7%	670,4	676,8	1,5%	677,2	691,32	3,6%
50	3	638,6	657,6	659,92	3,3%	657,8	659,76	3,3%	664,6	671,76	5,2%
50	3	596	596,6	603,88	1,3%	600,6	603	1,2%	599,4	607,6	1,9%
50	3	629,8	637	644,72	2,4%	629,8	640,48	1,7%	640	649,48	3,1%
50	3	680,2	682,2	690,04	1,4%	680,2	685,52	0,8%	688,6	693,44	1,9%
50	3	591,8	592,6	597,8	1,0%	594,6	597,88	1,0%	595,8	612,6	3,5%
50	3	587,2	595,4	599,64	2,1%	595,6	596,76	1,6%	612,6	618,04	5,3%

n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie	Min	Avarage	Ie
50	3	631	635,8	642,56	1,8%	632,2	634,48	0,6%	642	650,4	3,1%
50	3	680	687,4	690,56	1,6%	683,8	689,68	1,4%	701,6	708	4,1%
50	3	609,6	614,6	621,52	2,0%	618,6	628,52	3,1%	634,6	648,6	6,4%
50	3	584,8	592,2	597,28	2,1%	584,8	594,96	1,7%	604,8	617,48	5,6%
50	3	568,8	573,8	578,28	1,7%	575,2	582,76	2,5%	592,6	598,08	5,1%
50	3	506,4	509,4	514	1,5%	506,4	510,72	0,9%	515,2	526,92	4,1%
50	3	559,2	559,2	561,6	0,4%	565,4	568,24	1,6%	571,4	583,48	4,3%
50	3	595,6	596,8	601,96	1,1%	596,2	604,92	1,6%	609	615,84	3,4%
50	3	525	525,8	536	2,1%	537,8	544,72	3,8%	554,2	564,2	7,5%
50	3	540,6	547	556,32	2,9%	554,6	557,8	3,2%	565,4	579,04	7,1%
50	3	584,6	596,4	601,12	2,8%	584,6	594,92	1,8%	613,2	625,68	7,0%
50	3	663	663	673,92	1,6%	665,8	672,32	1,4%	691,2	698,76	5,4%
50	3	629,8	629,8	635,48	0,9%	635,4	638,8	1,4%	641,2	654,24	3,9%
50	3	572,4	572,4	578,6	1,1%	575,2	582,76	1,8%	583,8	600,48	4,9%
50	4	484,6	485,4	489,48	1,0%	484,6	489,56	1,0%	495,8	498,8	2,9%
50	4	512	516,6	518,64	1,3%	512	518,68	1,3%	523,6	529,2	3,4%
50	4	557,6	558,2	561,56	0,7%	559,2	560,92	0,6%	573,2	579,92	4,0%
50	4	544,4	544,4	551,28	1,3%	549,8	553,72	1,7%	559,4	568,16	4,4%
50	4	512,4	513,6	522,76	2,0%	515,6	518,36	1,2%	528,6	540,08	5,4%
50	4	502,4	504,4	506,8	0,9%	505,8	507,44	1,0%	511,2	517,88	3,1%
50	4	500,8	500,8	504,04	0,6%	501,8	505,2	0,9%	515,6	525,92	5,0%
50	4	358	360	362,36	1,2%	358	362,76	1,3%	361,6	375,56	4,9%
50	4	435	439	442,6	1,7%	443	446,16	2,6%	452,2	456,36	4,9%
50	4	445,8	452,6	458,4	2,8%	451,8	455,88	2,3%	445,8	461,8	3,6%
50	4	421,6	426,8	428,92	1,7%	421,6	427,08	1,3%	441	445,96	5,8%
50	4	445,4	445,4	451,32	1,3%	448,2	449,84	1,0%	459,2	471,4	5,8%
50	4	501,4	511	516,52	3,0%	501,4	512,04	2,1%	528	544	8,5%
50	4	481,2	483,8	491,48	2,1%	481,2	486,76	1,2%	489,6	509,92	6,0%
50	4	444,4	444,4	451,96	1,7%	453,8	456,76	2,8%	460,2	467,56	5,2%
50	4	450,8	452,8	455,4	1,0%	450,8	459,48	1,9%	472	479,24	6,3%
50	4	441,4	441,4	448,12	1,5%	443,2	447,64	1,4%	463	473,68	7,3%
50	4	317,4	325,2	328,36	3,5%	321,2	325,48	2,5%	336,4	347,24	9,4%
50	4	377,8	380,2	382,96	1,4%	377,8	383,56	1,5%	400,8	406,76	7,7%
50	4	403	404,2	405,84	0,7%	403	408,76	1,4%	412,8	430,52	6,8%
50	4	395,4	396,8	401,44	1,5%	397	402,4	1,8%	415,8	423,24	7,0%
50	4	428,2	432,4	434,88	1,6%	428,2	432,24	0,9%	445,4	453,16	5,8%
50	4	476,2	482,4	487,04	2,3%	483	488,12	2,5%	501,4	513,96	7,9%
50	4	451,4	455	460,48	2,0%	451,4	456,48	1,1%	472,8	483,64	7,1%
50	4	398,6	401,8	406	1,9%	404,8	407,56	2,2%	418,4	427,92	7,4%

Ie	5,33%	Iterations	249 477	Ie	1,87%	Iterations	455 035	Ie	1,80%	Iterations	450 762
Variance	0,00053	Time (s)	18,2	Variance	0,00026	Time (s)	32,2	Variance	0,00025	Time (s)	32,1
15,2	5,13%		227 759	15,2	2,39%		448 003	15,2	2,30%		429 347
	0,00082		6,0		0,00069		11,4		0,00067		10,9
20,2	4,80%		246 872	20,2	2,06%		445 944	20,2	2,01%		451 113
	0,00037		8,3		0,00025		14,5		0,00029		14,7
20,3	5,90%		258 910	20,3	2,00%		469 566	20,3	1,91%		460 919
	0,00040		12,0		0,00024		21,4		0,00017		21,0
50,3	4,74%		243 691	50,3	1,49%		431 164	50,3	1,42%		437 354
	0,00040		26,6		0,00004		46,5		0,00004		47,3
50,4	6,05%		270 155	50,4	1,42%		480 495	50,4	1,39%		475 074
	0,00055		37,9		0,00003		67,4		0,00003		66,7

			RSAI			**ESASI**			**RSASI**		
n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie	Min	Avarage	Ie
15	2	393,6	406,6	412	4,7%	396,2	397,16	0,9%	393,6	397,68	1,0%
15	2	331,4	338	339,24	2,4%	331,4	336,56	1,6%	332,8	337,12	1,7%
15	2	302,8	315,6	324,52	7,2%	302,8	308,08	1,7%	304,8	311,16	2,8%
15	2	439,4	499,4	507,36	15,5%	439,4	478,52	8,9%	476	480,56	9,4%
15	2	348,6	348,6	388,2	11,4%	375,6	383,44	10,0%	377	383,36	10,0%
15	2	384,4	395,8	405,08	5,4%	385,4	391,04	1,7%	391,2	392,12	2,0%
15	2	342,8	347,4	356,08	3,9%	342,8	348,16	1,6%	346,4	349,76	2,0%
15	2	390,4	398,8	403,36	3,3%	390,4	395,88	1,4%	393,8	396,88	1,7%
15	2	396,6	407	414,76	4,6%	396,6	399,12	0,6%	397,8	402,32	1,4%
15	2	337	346,8	357,72	6,1%	346	349,72	3,8%	337	347,88	3,2%
15	2	370	377,6	390,28	5,5%	371,8	376,52	1,8%	370,6	375,52	1,5%
15	2	285,4	314,6	322,6	13,0%	314,8	318,36	11,5%	314,6	321,24	12,6%

n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie	Min	Avarage	Ie
15	2	290,6	300,4	306,2	5,4%	295,6	296,72	2,1%	290,6	294,4	1,3%
15	2	440,6	457	466,52	5,9%	440,6	443,96	0,8%	440,6	446,35	1,3%
15	2	346,8	372,8	388,8	12,1%	371,8	375,84	8,4%	371,8	375,48	8,3%
15	2	350,8	351,8	365,12	4,1%	350,8	357,88	2,0%	351,8	355,8	1,4%
15	2	320,4	330,4	336,12	4,9%	324,8	328,2	2,4%	326,8	330,12	3,0%
15	2	357,4	368,4	375,84	5,2%	357,4	363,12	1,6%	357,4	361,04	1,0%
15	2	391,6	396,8	403,84	3,1%	394,2	395,28	0,9%	394,4	394,72	0,8%
15	2	312	320	324,84	4,1%	312,6	314,2	0,7%	312,6	316,56	1,5%
15	2	347,2	359,4	366,04	5,4%	347,2	350,08	0,8%	347,2	349,92	0,8%
15	2	290,2	314	321,36	10,7%	306,8	310,56	7,0%	290,2	306,92	5,8%
15	2	300,2	301,4	310,76	3,5%	300,2	304,12	1,3%	305,4	307,04	2,3%
15	2	400,4	415,4	425,72	6,3%	400,6	405	1,1%	400,4	402,56	0,5%
15	2	382,4	391,4	402,52	5,3%	382,4	383,16	0,2%	382,4	384,4	0,5%
20	2	337,6	342	349,72	3,6%	338	347	2,8%	341,4	346,36	2,6%
20	2	434	444,4	450,8	3,9%	438	441,6	1,8%	434	438	0,9%
20	2	372,2	381,8	385,56	3,6%	374,2	376,84	1,2%	373,8	376,28	1,1%
20	2	392,6	406	414,88	5,7%	401,6	402,68	2,6%	393,6	400,8	2,1%
20	2	451,2	459,2	470,16	4,2%	456,6	458,8	1,7%	452	457	1,3%
20	2	323,6	332,6	341	5,4%	323,6	331,56	2,5%	327,4	333,24	3,0%
20	2	472,8	486,2	495,48	4,8%	472,8	478,08	1,1%	476,8	478,2	1,1%
20	2	367,2	375,2	383,08	4,3%	373	374,32	1,9%	367,2	376,08	2,4%
20	2	423,8	436,4	443,16	4,6%	429,4	432,88	2,1%	430,4	433,72	2,3%
20	2	440,2	463,6	470	6,8%	440,2	456,84	3,8%	458,2	461,2	4,8%
20	2	291,4	291,4	321,56	10,4%	315	320,36	9,9%	321,8	327,36	12,3%
20	2	399,2	405,6	413,4	3,6%	401,6	404,8	1,4%	399,6	403,28	1,0%
20	2	328	338,6	351,72	7,2%	331	335,36	2,2%	328	334,2	1,9%
20	2	356	371,6	383,72	7,8%	356	365,76	2,7%	362,6	367,32	3,2%
20	2	398,4	423,8	428,24	7,5%	414	416,6	4,6%	412,6	417,72	4,8%
20	2	297,2	301,4	311,16	4,7%	298,4	300,76	1,2%	297,2	300,16	1,0%
20	2	425	438,4	448,48	5,5%	426,4	430,6	1,3%	429	430,52	1,3%
20	2	359,4	365,8	373,84	4,0%	359,4	363,76	1,2%	359,4	362,6	0,9%
20	2	414,2	418,8	430,56	3,9%	419,6	421,56	1,8%	417,2	422,04	1,9%
20	2	450,4	455,4	459,44	2,0%	450,4	455,68	1,2%	450,4	453,88	0,8%
20	2	316,4	321,6	327,48	3,5%	316,4	321,04	1,5%	321,4	324,64	2,6%
20	2	393,6	396,8	407,68	3,6%	397,8	399,6	1,5%	398,2	401,36	2,0%
20	2	308	308,4	325,52	5,7%	310,2	312,64	1,5%	308	310,24	0,7%
20	2	365,8	370,8	377,92	3,3%	367,4	370,92	1,4%	369,6	371,48	1,6%
20	2	409,4	423,4	425,88	4,0%	409,4	414,68	1,3%	411,4	414,28	1,2%
20	3	357,6	371,8	380,28	6,3%	361,4	363,96	1,8%	357,6	364,52	1,9%
20	3	225,8	234,2	239,24	6,0%	225,8	235,76	4,4%	232,4	235,52	4,3%
20	3	364,4	372,2	384,16	5,4%	369,6	371,72	2,0%	369,4	371,76	2,0%
20	3	309,8	327	335,44	8,3%	309,8	315,6	1,9%	313,4	319,4	3,1%
20	3	263,4	267,2	276	4,8%	267,2	270,44	2,7%	268,2	269,16	2,2%
20	3	312,8	323,4	330,08	5,5%	317,6	321,76	2,9%	317,4	320,4	2,4%
20	3	305,8	314,6	323,72	5,9%	305,8	311,92	2,0%	310,4	311,24	1,8%
20	3	265,6	278,4	280,72	5,7%	271,6	272,88	2,7%	265,8	271,2	2,1%
20	3	287,2	306,2	317,12	10,4%	304,4	307,24	7,0%	304	306,4	6,7%
20	3	234,8	245	247,84	5,6%	240,8	242,84	3,4%	234,8	241,32	2,8%
20	3	344,8	357,8	362,88	5,2%	346,4	350,24	1,6%	344,8	350,36	1,6%
20	3	225	225	235,36	4,6%	227,4	232,16	3,2%	230,4	232,32	3,3%
20	3	330,8	347	359,8	8,8%	337,2	341,32	3,2%	336,8	341,52	3,2%
20	3	282,2	288,4	294,92	4,5%	284,6	286,44	1,5%	285,6	288,12	2,1%
20	3	240,6	256,6	260,56	8,3%	240,6	244,36	1,6%	247,2	248,92	3,5%
20	3	297	302,4	314,28	5,8%	297	303,2	2,1%	300,2	303,24	2,1%
20	3	292	297,2	309,84	6,1%	292,2	296,2	1,4%	292	299,4	2,5%
20	3	246,2	253,6	258,72	5,1%	246,8	251,16	2,0%	246,2	248,36	0,9%
20	3	281,6	291,6	298,24	5,9%	286,4	290,48	3,2%	281,6	289,92	3,0%
20	3	218,2	228,8	234,2	7,3%	221	225,92	3,5%	218,2	223,08	2,2%
20	3	337,2	343,4	351,64	4,3%	341,8	343,12	1,8%	341,4	343,2	1,8%
20	3	225,6	238,2	240,76	6,7%	225,6	231,32	2,5%	230,4	234,24	3,8%
20	3	316,2	333,6	340,48	7,7%	316,2	320,8	1,5%	317,8	320,44	1,3%
20	3	251,2	262,6	268,72	7,0%	251,2	255,72	1,8%	253,8	256,48	2,1%
20	3	226,2	233	239,68	6,0%	226,2	229,4	1,4%	228,6	232,2	2,7%
50	3	663	693,4	695,28	4,9%	668,8	671,12	1,2%	671,8	674,28	1,7%
50	3	704,8	717,4	729,16	3,5%	711,8	715,96	1,6%	705,2	717,04	1,7%
50	3	751	779,2	788,36	5,0%	756	764,24	1,8%	756,4	760,68	1,3%
50	3	686,6	711,2	721,12	5,0%	691,8	699,64	1,9%	686,6	697,36	1,6%

n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie	Min	Avarage	Ie
50	3	667	683,4	690,92	3,6%	668,4	674,56	1,1%	670,4	673,72	1,0%
50	3	638,6	638,6	668,64	4,7%	650	658,76	3,2%	649,4	655,76	2,7%
50	3	596	597,8	606,32	1,7%	597,2	604,24	1,4%	596	602,4	1,1%
50	3	629,8	644,6	651,16	3,4%	633,4	639,92	1,6%	633,4	640,28	1,7%
50	3	680,2	693,8	696,64	2,4%	680,8	689,32	1,3%	683	688,84	1,3%
50	3	591,8	598,8	607,48	2,6%	592,8	594,96	0,5%	591,8	594,44	0,4%
50	3	587,2	618,4	628,28	7,0%	595,2	599,72	2,1%	587,2	596,28	1,5%
50	3	631	649,6	656,68	4,1%	635,6	642,84	1,9%	631	640,12	1,4%
50	3	680	698,8	708,12	4,1%	683,8	690,28	1,5%	680	687,28	1,1%
50	3	609,6	640,4	653	7,1%	620,8	624,68	2,5%	609,6	621,96	2,0%
50	3	584,8	611,2	616,8	5,5%	587,8	596	1,9%	589	599,04	2,4%
50	3	568,8	583,2	591,16	3,9%	568,8	572,08	0,6%	572,8	577,04	1,4%
50	3	506,4	530	534,68	5,6%	508,2	513,04	1,3%	507,8	511,92	1,1%
50	3	559,2	566,8	572,84	2,4%	563,8	571,44	2,2%	563,8	569,2	1,8%
50	3	595,6	608,8	619,8	4,1%	595,6	600,04	0,7%	596,6	604,6	1,5%
50	3	525	557,6	565,28	7,7%	531	538,16	2,5%	525	539,68	2,8%
50	3	540,6	564,6	575,56	6,5%	540,6	554,68	2,6%	545,8	556,64	3,0%
50	3	584,6	611,6	616,44	5,4%	591,2	594,96	1,8%	586	594	1,6%
50	3	663	690	703,52	6,1%	667,6	673,12	1,5%	669,8	673,16	1,5%
50	3	629,8	641,4	668,2	6,1%	634	638,88	1,4%	633,2	639,36	1,5%
50	3	572,4	592	606,2	5,9%	577,8	582,8	1,8%	574,4	577,88	1,0%
50	4	484,6	492,8	504,12	4,0%	487,6	491,84	1,5%	485	490,48	1,2%
50	4	512	524,6	535,52	4,6%	516,4	517,8	1,1%	514,4	518,12	1,2%
50	4	557,6	570	574,84	3,1%	557,6	559,8	0,4%	559,2	562,4	0,9%
50	4	544,4	559,4	572,32	5,1%	549	552,08	1,4%	548,4	551,6	1,3%
50	4	512,4	526,2	539,2	5,2%	512,4	517,44	1,0%	519,6	520,6	1,6%
50	4	502,4	514	522,52	4,0%	502,4	506,36	0,8%	503,6	508,16	1,1%
50	4	500,8	510,6	520,2	3,9%	501,4	505,6	1,0%	503,6	507,52	1,3%
50	4	358	369,6	375,44	4,9%	358,8	361,08	0,9%	358	364,88	1,9%
50	4	435	447,6	454,44	4,5%	435	442,6	1,7%	439	442,92	1,8%
50	4	445,8	461,8	467,56	4,9%	450,4	455,24	2,1%	452,6	456,64	2,4%
50	4	421,6	443,8	448,84	6,5%	423,6	427,88	1,5%	424,4	429,24	1,8%
50	4	445,4	456,8	466,96	4,8%	449	451,52	1,4%	447,8	451,64	1,4%
50	4	501,4	520	532,52	6,2%	507,6	511,44	2,0%	508,8	513,4	2,4%
50	4	481,2	503,2	509,76	5,9%	484	489,32	1,7%	485	489,64	1,8%
50	4	444,4	461,4	464,92	4,6%	453,8	456,92	2,8%	448,4	453,76	2,1%
50	4	450,8	469,4	482,2	7,0%	451,8	457,92	1,6%	455,8	459,84	2,0%
50	4	441,4	461,6	465,76	5,5%	444,4	449,16	1,8%	445	449,96	1,9%
50	4	317,4	339,8	348,08	9,7%	321,4	327	3,0%	317,4	324,4	2,2%
50	4	377,8	393,6	400,84	6,1%	382,6	389,12	3,0%	384,6	386,68	2,4%
50	4	403	414,2	423,44	5,1%	403,2	408,8	1,4%	403	407,6	1,1%
50	4	395,4	421,4	434,32	9,8%	395,4	402,16	1,7%	397	400,64	1,3%
50	4	428,2	454,6	468	9,3%	431,6	434,92	1,6%	429,2	433,08	1,1%
50	4	476,2	509,4	519,36	9,1%	476,2	482,52	1,3%	480,2	483,76	1,6%
50	4	451,4	466,6	477,52	5,8%	453,8	458,6	1,6%	457,2	460,36	2,0%
50	4	398,6	418,8	426,76	7,1%	398,6	404,96	1,6%	402,6	405,12	1,6%

Phase 3: Bicriteria

For the complete results, see the archive Phase 3 Bicriteria.xls on the CD.

The results for α = 0.6 ESASI are demonstrated as an example together with the summary and the complete set of calculations for both ESASI and RSASI and α = 0.6 and α = 0.8.

alfa=0.6 ESASI

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	fbest total	Cmaxbest	Tmaxbest	Time	it
50	3	1	100	1,2	0,8	0,2	10	913	777,6	795,8	777,6	1005	482	42,71875	393193
50	3	2	100	1,2	0,8	0,2	10	948,4	819,4	845	819,4	1045	545	34,23438	316360
50	3	3	100	1,2	0,8	0,2	10	986,8	868,8	878,8	868,8	1090	562	20,71875	191258
50	3	4	100	1,2	0,8	0,2	10	911,4	804,8	831	804,8	1043	513	50,6875	466864
50	3	5	100	1,2	0,8	0,2	10	873,4	779,4	803,6	779,4	1020	479	43,98438	408838
50	3	6	100	1,2	0,8	0,2	10	874,8	770,2	777,4	770,2	993	454	39,125	361948
50	3	7	100	1,2	0,8	0,2	10	819,4	713,2	724,2	713,2	937	405	31	286796
50	3	8	100	1,2	0,8	0,2	10	867	751,6	764,6	751,6	949	488	29,07813	267039
50	3	9	100	1,2	0,8	0,2	10	914,8	793,2	804,6	793,2	1013	492	28,4375	263778
50	3	10	100	1,2	0,8	0,2	10	817,8	704,2	687,4	687,4	891	382	37,09375	342780
50	3	1	100	1,2	0,8	0,2	10	913	779	785	779	995	470	50,95313	470137
50	3	2	100	1,2	0,8	0,2	10	948,4	823,6	836,4	823,6	1050	516	57,07813	527119
50	3	3	100	1,2	0,8	0,2	10	986,8	865,2	887,8	865,2	1099	571	37,625	346187
50	3	4	100	1,2	0,8	0,2	10	911,4	812	822,6	812	1033	507	23,34375	215147
50	3	5	100	1,2	0,8	0,2	10	873,4	784,4	794,8	784,4	1002	484	30,32813	279855
50	3	6	100	1,2	0,8	0,2	10	874,8	765,2	775,8	765,2	991	453	47,59375	441364
50	3	7	100	1,2	0,8	0,2	10	819,4	717	708	708	922	387	28,875	267619
50	3	8	100	1,2	0,8	0,2	10	867	742	768,2	742	959	482	36,51563	338063
50	3	9	100	1,2	0,8	0,2	10	914,8	799,6	810,4	799,6	1016	502	34,98438	323087
50	3	10	100	1,2	0,8	0,2	10	817,8	707,4	719,2	707,4	928	406	46,90625	432687
50	3	1	100	1,2	0,8	0,2	10	913	771	813,4	771	1017	508	39,45313	364333
50	3	2	100	1,2	0,8	0,2	10	948,4	820,4	839,2	820,4	1038	541	39,14063	361362
50	3	3	100	1,2	0,8	0,2	10	986,8	874,8	894	874,8	1104	579	31,95313	295123
50	3	4	100	1,2	0,8	0,2	10	911,4	807	819,6	807	1030	504	44,71875	413348
50	3	5	100	1,2	0,8	0,2	10	873,4	787,8	792	787,8	1006	471	30,46875	281474
50	3	6	100	1,2	0,8	0,2	10	874,8	761,4	793,4	761,4	1007	473	47,84375	442998
50	3	7	100	1,2	0,8	0,2	10	819,4	716,2	717,8	716,2	933	395	31,60938	292971
50	3	8	100	1,2	0,8	0,2	10	867	736,8	759,6	736,8	956	465	44,73438	413009
50	3	9	100	1,2	0,8	0,2	10	914,8	803,6	808,2	803,6	1021	489	26,39063	242958
50	3	10	100	1,2	0,8	0,2	10	817,8	706,2	719,2	706,2	920	418	61,125	561683
50	3	1	100	1,2	0,8	0,2	10	913	784	798,8	784	1010	482	47,15625	434436
50	3	2	100	1,2	0,8	0,2	10	948,4	816,8	832	816,8	1040	520	41,53125	383853
50	3	3	100	1,2	0,8	0,2	10	986,8	875,4	906,8	875,4	1116	593	22,375	205991
50	3	4	100	1,2	0,8	0,2	10	911,4	817	830,6	817	1043	512	49,96875	459371
50	3	5	100	1,2	0,8	0,2	10	873,4	780,2	793,6	780,2	1008	472	43,40625	399219
50	3	6	100	1,2	0,8	0,2	10	874,8	763	774	763	986	456	37,59375	347254
50	3	7	100	1,2	0,8	0,2	10	819,4	705,6	713,2	705,6	926	394	36,98438	342504
50	3	8	100	1,2	0,8	0,2	10	867	738,4	747,2	738,4	950	443	37,29688	344406
50	3	9	100	1,2	0,8	0,2	10	914,8	793,8	807	793,8	1013	498	50,21875	462617
50	3	10	100	1,2	0,8	0,2	10	817,8	708,2	712,8	708,2	918	405	40,45313	371208
50	3	1	100	1,2	0,8	0,2	10	913	780,6	803,8	780,6	1011	493	43,01563	395101
50	3	2	100	1,2	0,8	0,2	10	948,4	819,6	837,6	819,6	1040	534	42,14063	389186
50	3	3	100	1,2	0,8	0,2	10	986,8	870,2	881,2	870,2	1098	556	49,75	461608
50	3	4	100	1,2	0,8	0,2	10	911,4	801	811	801	1023	493	51,28125	472050
50	3	5	100	1,2	0,8	0,2	10	873,4	786,8	798,6	786,8	1007	486	42,40625	392032
50	3	6	100	1,2	0,8	0,2	10	874,8	771,4	776,8	771,4	992	454	24,20313	224174
50	3	7	100	1,2	0,8	0,2	10	819,4	704,2	710,8	704,2	926	388	44,5625	413408
50	3	8	100	1,2	0,8	0,2	10	867	745,6	736	736	940	430	46,15625	426874
50	3	9	100	1,2	0,8	0,2	10	914,8	789,6	798,8	789,6	1008	485	44,07813	407698
50	3	10	100	1,2	0,8	0,2	10	817,8	695,6	711	695,6	915	405	45,125	418876
50	3	1	100	0,5	0,8	0,4	10	746,6	683,4	688,6	683,4	883	397	45,79688	422700
50	3	2	100	0,5	0,8	0,4	10	655,2	602,6	602,6	602,6	791	320	36,51563	335222
50	3	3	100	0,5	0,8	0,4	10	748,2	693,8	695,6	693,8	876	425	38,5	354455
50	3	4	100	0,5	0,8	0,4	10	834	750,8	750	750	944	459	43,51563	402520

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	fbest total	Cmaxbest	Tmaxbest	Time	it
50	3	5	100	0,5	0,8	0,4	10	889,6	813,6	831,6	813,6	1016	555	38,3125	352624
50	3	6	100	0,5	0,8	0,4	10	722,2	665,8	673	665,8	869	379	31,5	291472
50	3	7	100	0,5	0,8	0,4	10	757,4	717,4	739,2	717,4	882	525	31,20313	287924
50	3	8	100	0,5	0,8	0,4	10	814,8	731	741,4	731	929	460	53,54688	492625
50	3	9	100	0,5	0,8	0,4	10	910,2	800	798,2	798,2	993	506	27,92188	257098
50	3	10	100	0,5	0,8	0,4	10	840,8	786	790,8	786	976	513	42,70313	392328
50	3	1	100	0,5	0,8	0,4	10	746,6	685,2	699,2	685,2	896	404	52,04688	478956
50	3	2	100	0,5	0,8	0,4	10	655,2	598,8	608,6	598,8	803	317	39,53125	363994
50	3	3	100	0,5	0,8	0,4	10	748,2	691,8	694	691,8	880	415	42,03125	387563
50	3	4	100	0,5	0,8	0,4	10	834	742,6	753,8	742,6	947	464	41,95313	386133
50	3	5	100	0,5	0,8	0,4	10	889,6	812,8	817,6	812,8	1010	529	46,46875	427909
50	3	6	100	0,5	0,8	0,4	10	722,2	663,4	673,4	663,4	857	398	44,42188	411208
50	3	7	100	0,5	0,8	0,4	10	757,4	708	709,8	708	877	459	49,90625	460859
50	3	8	100	0,5	0,8	0,4	10	814,8	729,6	734,2	729,6	927	445	39,26563	362218
50	3	9	100	0,5	0,8	0,4	10	910,2	793,8	806,4	793,8	1000	516	45,875	423614
50	3	10	100	0,5	0,8	0,4	10	840,8	785	807,2	785	978	551	32,54688	300762
50	3	1	100	0,5	0,8	0,4	10	746,6	682,4	700	682,4	888	418	45,8125	424624
50	3	2	100	0,5	0,8	0,4	10	655,2	599,2	628,2	599,2	811	354	61,07813	565059
50	3	3	100	0,5	0,8	0,4	10	748,2	690,2	703,2	690,2	872	450	38,04688	351226
50	3	4	100	0,5	0,8	0,4	10	834	754	759,8	754	949	476	36,96875	340220
50	3	5	100	0,5	0,8	0,4	10	889,6	812,8	818,6	812,8	1015	524	44,32813	407803
50	3	6	100	0,5	0,8	0,4	10	722,2	667,6	674	667,6	864	389	37,85938	347826
50	3	7	100	0,5	0,8	0,4	10	757,4	714	731,6	714	884	503	32,35938	298722
50	3	8	100	0,5	0,8	0,4	10	814,8	732	730,6	730,6	923	442	33,85938	313051
50	3	9	100	0,5	0,8	0,4	10	910,2	794,4	801,6	794,4	996	510	49,79688	459932
50	3	10	100	0,5	0,8	0,4	10	840,8	778,6	800,6	778,6	977	536	37,76563	346656
50	3	1	100	0,5	0,8	0,4	10	746,6	690,8	693,4	690,8	887	403	32,89063	303256
50	3	2	100	0,5	0,8	0,4	10	655,2	611,2	613,6	611,2	806	325	41,23438	381891
50	3	3	100	0,5	0,8	0,4	10	748,2	690,2	698,2	690,2	883	421	43,57813	401962
50	3	4	100	0,5	0,8	0,4	10	834	752,4	756,8	752,4	944	476	33,25	306617
50	3	5	100	0,5	0,8	0,4	10	889,6	813,2	823,2	813,2	1012	540	21,95313	203521
50	3	6	100	0,5	0,8	0,4	10	722,2	663	674,4	663	852	408	41,35938	383501
50	3	7	100	0,5	0,8	0,4	10	757,4	705,4	722,4	705,4	892	468	47,39063	435656
50	3	8	100	0,5	0,8	0,4	10	814,8	729,8	734,2	729,8	927	445	55,3125	510377
50	3	9	100	0,5	0,8	0,4	10	910,2	796,2	801,2	796,2	996	509	39,29688	361970
50	3	10	100	0,5	0,8	0,4	10	840,8	786	792	786	980	510	38,64063	356169
50	3	1	100	0,5	0,8	0,4	10	746,6	690,2	693,2	690,2	884	407	26,04688	239780
50	3	2	100	0,5	0,8	0,4	10	655,2	604,8	609,4	604,8	805	316	43,32813	399964
50	3	3	100	0,5	0,8	0,4	10	748,2	700,6	700,4	700,4	878	434	24,60938	226102
50	3	4	100	0,5	0,8	0,4	10	834	750,4	752	750,4	944	464	39,3125	363162
50	3	5	100	0,5	0,8	0,4	10	889,6	816,6	830,6	816,6	1017	551	34,28125	315995
50	3	6	100	0,5	0,8	0,4	10	722,2	666,2	667	666,2	859	379	25,3125	233659
50	3	7	100	0,5	0,8	0,4	10	757,4	702,2	716	702,2	856	506	49,03125	454073
50	3	8	100	0,5	0,8	0,4	10	814,8	729,2	752,2	729,2	943	466	35,10938	324507
50	3	9	100	0,5	0,8	0,4	10	910,2	795	804	795	998	513	41,42188	381249
50	3	10	100	0,5	0,8	0,4	10	840,8	782	788	782	976	506	45,89063	423446
50	3	1	100	0,8	0,8	0,8	10	868,6	795,2	814,8	795,2	1036	483	45,71875	422376
50	3	2	100	0,8	0,8	0,8	10	644	594,8	584,4	584,4	844	195	43,34375	400501
50	3	3	100	0,8	0,8	0,8	10	719,2	678,2	746	678,2	972	407	62,875	578699
50	3	4	100	0,8	0,8	0,8	10	705,4	665	677	665	897	347	50,8125	467804
50	3	5	100	0,8	0,8	0,8	10	716	680	690,6	680	959	288	44,59375	413264
50	3	6	100	0,8	0,8	0,8	10	704,4	678,2	673,4	673,4	933	284	38,84375	358078
50	3	7	100	0,8	0,8	0,8	10	686,8	661,2	675,6	661,2	948	267	32,0625	295849
50	3	8	100	0,8	0,8	0,8	10	754	719,6	723	719,6	977	342	35,73438	331627
50	3	9	100	0,8	0,8	0,8	10	772	725,8	734,2	725,8	991	349	43,15625	398102
50	3	10	100	0,8	0,8	0,8	10	773,6	724,4	731,6	724,4	996	335	46,54688	429291
50	3	1	100	0,8	0,8	0,8	10	868,6	801,8	818,2	801,8	1055	463	33,17188	307121
50	3	2	100	0,8	0,8	0,8	10	644	591,8	594,8	591,8	860	197	59,28125	546871
50	3	3	100	0,8	0,8	0,8	10	719,2	669	691	669	927	337	40,57813	374629
50	3	4	100	0,8	0,8	0,8	10	705,4	658,6	688,6	658,6	895	379	45	414961
50	3	5	100	0,8	0,8	0,8	10	716	685,8	695,2	685,8	956	304	34,875	320743
50	3	6	100	0,8	0,8	0,8	10	704,4	670,8	683,2	670,8	928	316	41,67188	385441
50	3	7	100	0,8	0,8	0,8	10	686,8	672,2	676,2	672,2	955	258	19,21875	177356
50	3	8	100	0,8	0,8	0,8	10	754	718,6	745,4	718,6	977	398	44,6875	414489
50	3	9	100	0,8	0,8	0,8	10	772	735,4	749,2	735,4	1018	346	45,21875	416530
50	3	10	100	0,8	0,8	0,8	10	773,6	730,6	765	730,6	997	417	53,14063	490686

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	fbest total	Cmaxbest	Tmaxbest	Time	it
50	3	1	100	0,8	0,8	0,8	10	868,6	801	842,2	801	1039	547	49,95313	463690
50	3	2	100	0,8	0,8	0,8	10	644	596	592,8	592,8	830	237	56,98438	526998
50	3	3	100	0,8	0,8	0,8	10	719,2	672	679	672	925	310	43,76563	407019
50	3	4	100	0,8	0,8	0,8	10	705,4	662,8	665,2	662,8	900	313	42,20313	389585
50	3	5	100	0,8	0,8	0,8	10	716	675,2	682,2	675,2	945	288	50,4375	466234
50	3	6	100	0,8	0,8	0,8	10	704,4	688,8	676,2	676,2	935	288	33,51563	309087
50	3	7	100	0,8	0,8	0,8	10	686,8	664	673,6	664	964	238	63,15625	585061
50	3	8	100	0,8	0,8	0,8	10	754	713,6	731,8	713,6	975	367	64,9375	599275
50	3	9	100	0,8	0,8	0,8	10	772	725,8	763,4	725,8	1001	407	34,75	321313
50	3	10	100	0,8	0,8	0,8	10	773,6	721,6	752,8	721,6	1038	325	31,78125	293280
50	3	1	100	0,8	0,8	0,8	10	868,6	806,2	819,6	806,2	1048	477	42,82813	395751
50	3	2	100	0,8	0,8	0,8	10	644	591,8	603,6	591,8	856	225	28,54688	263059
50	3	3	100	0,8	0,8	0,8	10	719,2	692,2	702	692,2	940	345	47,125	434530
50	3	4	100	0,8	0,8	0,8	10	705,4	660,8	689	660,8	887	392	61,35938	567225
50	3	5	100	0,8	0,8	0,8	10	716	684	697,2	684	962	300	50,35938	462050
50	3	6	100	0,8	0,8	0,8	10	704,4	668,8	695,8	668,8	933	340	42,90625	396015
50	3	7	100	0,8	0,8	0,8	10	686,8	655,8	676,8	655,8	946	273	43,75	403544
50	3	8	100	0,8	0,8	0,8	10	754	720,4	729,4	720,4	977	358	46,1875	427286
50	3	9	100	0,8	0,8	0,8	10	772	733,4	743,6	733,4	1012	341	44,5625	410847
50	3	10	100	0,8	0,8	0,8	10	773,6	742,8	754,4	742,8	996	392	37,32813	345228
50	3	1	100	0,8	0,8	0,8	10	868,6	803,8	808,4	803,8	1036	467	47,5625	440197
50	3	2	100	0,8	0,8	0,8	10	644	578,4	593,8	578,4	845	217	51,4375	476295
50	3	3	100	0,8	0,8	0,8	10	719,2	679	697,4	679	931	347	57,70313	535277
50	3	4	100	0,8	0,8	0,8	10	705,4	651,2	689	651,2	885	395	27,8125	257382
50	3	5	100	0,8	0,8	0,8	10	716	677	700,8	677	960	312	63,95313	589490
50	3	6	100	0,8	0,8	0,8	10	704,4	680,6	702,8	680,6	930	362	43	397662
50	3	7	100	0,8	0,8	0,8	10	686,8	662,4	672,4	662,4	958	244	39,125	359323
50	3	8	100	0,8	0,8	0,8	10	754	717	718,6	717	969	343	52,29688	482992
50	3	9	100	0,8	0,8	0,8	10	772	719	719	719	989	314	47,60938	440887
50	3	10	100	0,8	0,8	0,8	10	773,6	726,6	742,4	726,6	996	362	48,20313	445029
50	4	1	100	1,2	0,8	0,2	10	648,2	574	574,8	574	732	339	48,625	345959
50	4	2	100	1,2	0,8	0,2	10	655,6	598	605,8	598	763	370	40,04688	284540
50	4	3	100	1,2	0,8	0,2	10	708	644,6	649,8	644,6	797	429	45,90625	326270
50	4	4	100	1,2	0,8	0,2	10	706,6	626	662,4	626	814	435	66,51563	472589
50	4	5	100	1,2	0,8	0,2	10	697,2	601,8	602,2	601,8	761	364	64,79688	461211
50	4	6	100	1,2	0,8	0,2	10	701,2	582,8	599,8	582,8	737	394	64,26563	456891
50	4	7	100	1,2	0,8	0,2	10	658,2	593,2	600	593,2	752	372	62,42188	443930
50	4	8	100	1,2	0,8	0,2	10	500,2	449,8	446	446	598	218	64,60938	459741
50	4	9	100	1,2	0,8	0,2	10	590,6	522	535,6	522	692	301	43,53125	309868
50	4	10	100	1,2	0,8	0,2	10	611,4	532,6	539,8	532,6	697	304	68,28125	485395
50	4	1	100	1,2	0,8	0,2	10	648,2	569,2	589,4	569,2	745	356	68,23438	485285
50	4	2	100	1,2	0,8	0,2	10	655,6	597,4	618,4	597,4	778	379	56,82813	404238
50	4	3	100	1,2	0,8	0,2	10	708	642,8	669,4	642,8	831	427	64,3125	457344
50	4	4	100	1,2	0,8	0,2	10	706,6	630,2	655,6	630,2	810	424	49,23438	350589
50	4	5	100	1,2	0,8	0,2	10	697,2	605	612,2	605	771	374	51,14063	363395
50	4	6	100	1,2	0,8	0,2	10	701,2	583,6	606,8	583,6	754	386	48,42188	344676
50	4	7	100	1,2	0,8	0,2	10	658,2	590,8	611,6	590,8	764	383	60,26563	428773
50	4	8	100	1,2	0,8	0,2	10	500,2	443,8	458,8	443,8	610	232	50,85938	361309
50	4	9	100	1,2	0,8	0,2	10	590,6	517	534,8	517	682	314	41,67188	296250
50	4	10	100	1,2	0,8	0,2	10	611,4	533,6	540,2	533,6	693	311	53,35938	379006
50	4	1	100	1,2	0,8	0,2	10	648,2	572,6	587,2	572,6	748	346	52,82813	375311
50	4	2	100	1,2	0,8	0,2	10	655,6	601,4	602,4	601,4	764	360	80,5625	572893
50	4	3	100	1,2	0,8	0,2	10	708	638,8	665,2	638,8	826	424	78,78125	559833
50	4	4	100	1,2	0,8	0,2	10	706,6	628,2	650,4	628,2	804	420	63,85938	454006
50	4	5	100	1,2	0,8	0,2	10	697,2	604	614,4	604	770	381	56,90625	404769
50	4	6	100	1,2	0,8	0,2	10	701,2	582,8	600,6	582,8	741	390	61,53125	437322
50	4	7	100	1,2	0,8	0,2	10	658,2	589,4	592,6	589,4	749	358	68,21875	484972
50	4	8	100	1,2	0,8	0,2	10	500,2	445,4	445,4	445,4	585	236	45,78125	325941
50	4	9	100	1,2	0,8	0,2	10	590,6	525,2	531,4	525,2	685	301	46,53125	330856
50	4	10	100	1,2	0,8	0,2	10	611,4	545,4	551,6	545,4	704	323	46,53125	330200
50	4	1	100	1,2	0,8	0,2	10	648,2	564	571,2	564	732	330	52,32813	372029
50	4	2	100	1,2	0,8	0,2	10	655,6	598,2	603,4	598,2	761	367	57,29688	407880
50	4	3	100	1,2	0,8	0,2	10	708	640,4	659,8	640,4	815	427	59,67188	423788
50	4	4	100	1,2	0,8	0,2	10	706,6	634	651	634	809	414	71,6875	509619
50	4	5	100	1,2	0,8	0,2	10	697,2	596	599,8	596	759	361	48,96875	348823
50	4	6	100	1,2	0,8	0,2	10	701,2	586,2	596,6	586,2	739	383	47,03125	334720

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	fbest total	Cmaxbest	Tmaxbest	Time	it
50	4	7	100	1,2	0,8	0,2	10	658,2	593,6	604	593,6	754	379	54,15625	384881
50	4	8	100	1,2	0,8	0,2	10	500,2	439,2	452,2	439,2	605	223	52,78125	375586
50	4	9	100	1,2	0,8	0,2	10	590,6	520,2	528	520,2	682	297	50,21875	357161
50	4	10	100	1,2	0,8	0,2	10	611,4	534,4	542,4	534,4	696	312	60,8125	432650
50	4	1	100	1,2	0,8	0,2	10	648,2	569	583,6	569	732	361	47,875	340376
50	4	2	100	1,2	0,8	0,2	10	655,6	602,6	616,4	602,6	776	377	78,92188	560834
50	4	3	100	1,2	0,8	0,2	10	708	649,4	659,2	649,4	804	442	44,84375	318503
50	4	4	100	1,2	0,8	0,2	10	706,6	630,2	652	630,2	810	415	50,45313	358946
50	4	5	100	1,2	0,8	0,2	10	697,2	605	612	605	770	375	55,60938	395353
50	4	6	100	1,2	0,8	0,2	10	701,2	581,2	590,6	581,2	741	365	47,375	336622
50	4	7	100	1,2	0,8	0,2	10	658,2	586,2	615	586,2	761	396	43,17188	307135
50	4	8	100	1,2	0,8	0,2	10	500,2	437	461,8	437	607	244	65,98438	469213
50	4	9	100	1,2	0,8	0,2	10	590,6	518	523,4	518	675	296	54,98438	391781
50	4	10	100	1,2	0,8	0,2	10	611,4	533,2	541,4	533,2	695	311	52,71875	375151
50	4	1	100	0,5	0,8	0,4	10	672,2	604	634	604	754	454	58,32813	414439
50	4	2	100	0,5	0,8	0,4	10	514,2	459,6	471,6	459,6	610	264	64,45313	458170
50	4	3	100	0,5	0,8	0,4	10	572,4	518,8	520,4	518,8	666	302	66,3125	471892
50	4	4	100	0,5	0,8	0,4	10	551,4	494,2	510,4	494,2	636	322	68,9375	489798
50	4	5	100	0,5	0,8	0,4	10	596	533,2	539,4	533,2	685	321	58,34375	415169
50	4	6	100	0,5	0,8	0,4	10	585,4	537,8	549,6	537,8	670	369	38,85938	276924
50	4	7	100	0,5	0,8	0,4	10	611,8	535	552,8	535	692	344	54,67188	388285
50	4	8	100	0,5	0,8	0,4	10	606,8	547,8	566,2	547,8	707	355	51,9375	370035
50	4	9	100	0,5	0,8	0,4	10	632	576,4	582,8	576,4	720	377	69,98438	498017
50	4	10	100	0,5	0,8	0,4	10	624,6	571,6	586,6	571,6	725	379	44,70313	317660
50	4	1	100	0,5	0,8	0,4	10	672,2	605,8	620,8	605,8	754	421	56,70313	403691
50	4	2	100	0,5	0,8	0,4	10	514,2	459,6	470,8	459,6	606	268	61,03125	433525
50	4	3	100	0,5	0,8	0,4	10	572,4	519,6	537,2	519,6	666	344	98,84375	703100
50	4	4	100	0,5	0,8	0,4	10	551,4	501	504,6	501	635	309	51,29688	364493
50	4	5	100	0,5	0,8	0,4	10	596	529,6	556,4	529,6	680	371	49,34375	350975
50	4	6	100	0,5	0,8	0,4	10	585,4	537	548,4	537	670	366	62,70313	445737
50	4	7	100	0,5	0,8	0,4	10	611,8	537,8	556,6	537,8	695	349	68,42188	486585
50	4	8	100	0,5	0,8	0,4	10	606,8	554	562,6	554	707	346	43,79688	311459
50	4	9	100	0,5	0,8	0,4	10	632	575,8	586	575,8	716	391	62,03125	440361
50	4	10	100	0,5	0,8	0,4	10	624,6	574,4	579,4	574,4	723	364	55,03125	391094
50	4	1	100	0,5	0,8	0,4	10	672,2	610	606,6	606,6	753	387	60,875	432914
50	4	2	100	0,5	0,8	0,4	10	514,2	465	467,8	465	607	259	59,21875	421743
50	4	3	100	0,5	0,8	0,4	10	572,4	516	547,4	516	693	329	55,76563	396226
50	4	4	100	0,5	0,8	0,4	10	551,4	499	499	499	631	301	71,84375	510157
50	4	5	100	0,5	0,8	0,4	10	596	530,8	555	530,8	681	366	54,9375	390958
50	4	6	100	0,5	0,8	0,4	10	585,4	533	551,6	533	674	368	64,98438	462302
50	4	7	100	0,5	0,8	0,4	10	611,8	532,2	550,2	532,2	677	360	51,28125	364881
50	4	8	100	0,5	0,8	0,4	10	606,8	549,2	567,6	549,2	704	363	77,89063	553560
50	4	9	100	0,5	0,8	0,4	10	632	577,8	581,8	577,8	719	376	59,25	420873
50	4	10	100	0,5	0,8	0,4	10	624,6	576,2	580,6	576,2	723	367	57,28125	406915
50	4	1	100	0,5	0,8	0,4	10	672,2	599,2	619,8	599,2	753	420	54,53125	388023
50	4	2	100	0,5	0,8	0,4	10	514,2	465,4	461	461	601	251	43,51563	308868
50	4	3	100	0,5	0,8	0,4	10	572,4	519,8	525,6	519,8	670	309	69,1875	492373
50	4	4	100	0,5	0,8	0,4	10	551,4	498,4	513,2	498,4	636	329	53,17188	378050
50	4	5	100	0,5	0,8	0,4	10	596	533,4	541,8	533,4	687	324	48,54688	345203
50	4	6	100	0,5	0,8	0,4	10	585,4	533,8	545,2	533,8	674	352	61,34375	436483
50	4	7	100	0,5	0,8	0,4	10	611,8	537,4	548,8	537,4	688	340	57,3125	407475
50	4	8	100	0,5	0,8	0,4	10	606,8	552,8	570,4	552,8	712	358	87,03125	618689
50	4	9	100	0,5	0,8	0,4	10	632	578	589,8	578	721	393	65,46875	465208
50	4	10	100	0,5	0,8	0,4	10	624,6	571,4	593	571,4	733	383	51,6875	367433
50	4	1	100	0,5	0,8	0,4	10	672,2	603	603	603	749	384	76,46875	543403
50	4	2	100	0,5	0,8	0,4	10	514,2	463,4	470,8	463,4	608	265	64,1875	455827
50	4	3	100	0,5	0,8	0,4	10	572,4	518,2	548,2	518,2	669	367	58,5	416484
50	4	4	100	0,5	0,8	0,4	10	551,4	495,2	512,2	495,2	631	334	92,98438	659935
50	4	5	100	0,5	0,8	0,4	10	596	531,8	556,8	531,8	684	366	56,89063	404167
50	4	6	100	0,5	0,8	0,4	10	585,4	528,8	545,4	528,8	671	357	66,95313	476524
50	4	7	100	0,5	0,8	0,4	10	611,8	531,8	556,6	531,8	683	367	52,17188	371367
50	4	8	100	0,5	0,8	0,4	10	606,8	553,8	561	553,8	707	342	62,39063	443332
50	4	9	100	0,5	0,8	0,4	10	632	580,4	587,6	580,4	722	386	62,48438	443616
50	4	10	100	0,5	0,8	0,4	10	624,6	577,4	577,6	577,4	722	361	62,78125	446313
50	4	1	100	0,8	0,8	0,8	10	594	536,6	545	536,6	759	224	71,28125	507124
50	4	2	100	0,8	0,8	0,8	10	613,4	569	580,8	569	786	273	49,04688	349459

n	m	ex	p1	p2	p3	p4	p5	finit	fbesthalf	fbest	fbest total	Cmaxbest	Tmaxbest	Time	it
50	4	3	100	0,8	0,8	0,8	10	677	613,2	631	613,2	835	325	76,67188	545027
50	4	4	100	0,8	0,8	0,8	10	622,4	580,8	604	580,8	764	364	54,03125	384831
50	4	5	100	0,8	0,8	0,8	10	579,4	551	564,6	551	751	285	88,03125	626415
50	4	6	100	0,8	0,8	0,8	10	575,8	538,2	541,2	538,2	732	255	53,04688	377717
50	4	7	100	0,8	0,8	0,8	10	513,2	476,2	489,6	476,2	696	180	45,32813	323153
50	4	8	100	0,8	0,8	0,8	10	581,6	516,2	529,2	516,2	722	240	59,39063	422967
50	4	9	100	0,8	0,8	0,8	10	622,2	553,4	565,6	553,4	756	280	73,6875	524962
50	4	10	100	0,8	0,8	0,8	10	558,2	521,2	537,2	521,2	672	335	49,35938	351175
50	4	1	100	0,8	0,8	0,8	10	594	525,8	551,2	525,8	746	259	48,78125	347327
50	4	2	100	0,8	0,8	0,8	10	613,4	558,8	576	558,8	786	261	67,9375	483169
50	4	3	100	0,8	0,8	0,8	10	677	620,6	641,2	620,6	818	376	40,60938	289062
50	4	4	100	0,8	0,8	0,8	10	622,4	573,4	593,4	573,4	763	339	80,04688	569584
50	4	5	100	0,8	0,8	0,8	10	579,4	545,8	581,6	545,8	740	344	73,26563	522043
50	4	6	100	0,8	0,8	0,8	10	575,8	547,8	558,2	547,8	733	296	70,3125	500279
50	4	7	100	0,8	0,8	0,8	10	513,2	468,4	486	468,4	690	180	61,15625	435831
50	4	8	100	0,8	0,8	0,8	10	581,6	519,6	543	519,6	729	264	37,07813	263979
50	4	9	100	0,8	0,8	0,8	10	622,2	560	589,6	560	746	355	84,85938	605041
50	4	10	100	0,8	0,8	0,8	10	558,2	507	555	507	687	357	70,40625	501820
50	4	1	100	0,8	0,8	0,8	10	594	530,4	548,6	530,4	745	254	79	562218
50	4	2	100	0,8	0,8	0,8	10	613,4	567,2	575,6	567,2	772	281	60,9375	433492
50	4	3	100	0,8	0,8	0,8	10	677	622,2	630,4	622,2	818	349	67,92188	483161
50	4	4	100	0,8	0,8	0,8	10	622,4	577,6	601	577,6	771	346	77,59375	551804
50	4	5	100	0,8	0,8	0,8	10	579,4	550,2	564,8	550,2	744	296	63,0625	448031
50	4	6	100	0,8	0,8	0,8	10	575,8	545,6	593,4	545,6	737	378	63,45313	451037
50	4	7	100	0,8	0,8	0,8	10	513,2	475,6	506	475,6	692	227	73,875	526337
50	4	8	100	0,8	0,8	0,8	10	581,6	517,2	526,2	517,2	719	237	50,17188	356752
50	4	9	100	0,8	0,8	0,8	10	622,2	555,4	574,6	555,4	749	313	61,59375	438545
50	4	10	100	0,8	0,8	0,8	10	558,2	509,6	532,4	509,6	682	308	61,53125	438006
50	4	1	100	0,8	0,8	0,8	10	594	525,6	549,2	525,6	748	251	68,64063	488892
50	4	2	100	0,8	0,8	0,8	10	613,4	561,8	566	561,8	776	251	56	398443
50	4	3	100	0,8	0,8	0,8	10	677	614,4	656,4	614,4	820	411	66,9375	476916
50	4	4	100	0,8	0,8	0,8	10	622,4	576,4	615,4	576,4	769	385	70,8125	504887
50	4	5	100	0,8	0,8	0,8	10	579,4	544,2	544,2	544,2	731	264	74,79688	532605
50	4	6	100	0,8	0,8	0,8	10	575,8	538,6	538,6	538,6	727	256	66,20313	471155
50	4	7	100	0,8	0,8	0,8	10	513,2	472,6	519,2	472,6	710	233	75,76563	539479
50	4	8	100	0,8	0,8	0,8	10	581,6	515,6	551	515,6	737	272	71,90625	511589
50	4	9	100	0,8	0,8	0,8	10	622,2	556,4	581,2	556,4	758	316	60,53125	431174
50	4	10	100	0,8	0,8	0,8	10	558,2	517,8	530,8	517,8	670	322	54,98438	391090
50	4	1	100	0,8	0,8	0,8	10	594	532,2	538,6	532,2	735	244	59,15625	420810
50	4	2	100	0,8	0,8	0,8	10	613,4	563,2	585,4	563,2	787	283	47,01563	334402
50	4	3	100	0,8	0,8	0,8	10	677	623,4	669,4	623,4	825	436	71,15625	507085
50	4	4	100	0,8	0,8	0,8	10	622,4	573,4	602	573,4	764	359	66,20313	470822
50	4	5	100	0,8	0,8	0,8	10	579,4	551,6	582,6	551,6	743	342	66,46875	472156
50	4	6	100	0,8	0,8	0,8	10	575,8	538,6	550,8	538,6	730	282	71,04688	505658
50	4	7	100	0,8	0,8	0,8	10	513,2	474,8	518,8	474,8	696	253	61,3125	436754
50	4	8	100	0,8	0,8	0,8	10	581,6	517,2	525,6	517,2	712	246	67,54688	481768
50	4	9	100	0,8	0,8	0,8	10	622,2	563,2	588	563,2	752	342	60,40625	430424
50	4	10	100	0,8	0,8	0,8	10	558,2	519	532	519	686	301	54,70313	389293

Ie	1,10%	Iterations	407 074	Ie	1,03%	Iterations	411 809
Variance	0,00005	Time (s)	51,0	Variance	0,00003	Time (s)	51,6
50,3	1,22%		384 343	50,3	1,05%		391 369
	0,00009		41,6		0,00005		42,4
50,4	0,97%		429 806	50,4	1,01%		432 249
	0,00002		60,4		0,00001		60,8

			ESASI 0.6			**RSASI 0.6**		
n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie
50	3	766	771	778,44	1,62%	766	773,84	1,02%
50	3	814,8	816,8	819,96	0,63%	814,8	819,8	0,61%
50	3	863	865,2	870,88	0,91%	863	867,68	0,54%
50	3	801	801	808,36	0,92%	802	809,4	1,05%
50	3	779,2	779,4	783,72	0,58%	779,2	784,16	0,64%
50	3	760,2	761,4	766,24	0,79%	760,2	765,04	0,64%
50	3	677,4	704,2	709,44	4,73%	677,4	703,16	3,80%
50	3	736	736	740,96	0,67%	737	743,68	1,04%

n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie
50	3	789,6	789,6	795,96	0,81%	792,8	794,6	0,63%
50	3	687,4	687,4	700,96	1,97%	691,2	699,88	1,82%
50	3	682,4	682,4	686,4	0,59%	684,2	687,12	0,69%
50	3	595,6	598,8	603,32	1,30%	595,6	603,12	1,26%
50	3	690,2	690,2	693,28	0,45%	693	696,04	0,85%
50	3	742,6	742,6	749,88	0,98%	745	748,44	0,79%
50	3	811,2	812,8	813,8	0,32%	811,2	813,96	0,34%
50	3	659,2	663	665,2	0,91%	659,2	662,64	0,52%
50	3	702,2	702,2	709,4	1,03%	704,6	710	1,11%
50	3	728,6	729,2	730,04	0,20%	728,6	731,64	0,42%
50	3	791,4	793,8	795,52	0,52%	791,4	793,48	0,26%
50	3	778,6	778,6	783,52	0,63%	780,6	784,64	0,78%
50	3	794,2	795,2	801,6	0,93%	794,2	801,08	0,87%
50	3	578,4	578,4	587,84	1,63%	585,2	588,68	1,78%
50	3	661,8	669	678,08	2,46%	661,8	677,4	2,36%
50	3	647,4	651,2	659,68	1,90%	647,4	654,24	1,06%
50	3	675,2	675,2	680,4	0,77%	676,4	683,8	1,27%
50	3	667,6	668,8	673,96	0,95%	667,6	673,64	0,90%
50	3	649,6	655,8	663,12	2,08%	649,6	655,8	0,95%
50	3	711,4	713,6	717,84	0,91%	711,4	719,68	1,16%
50	3	719	719	727,88	1,24%	721,4	728,2	1,28%
50	3	706,8	721,6	729,2	3,17%	706,8	714,52	1,09%
50	4	564	564	569,76	1,02%	567,4	568,88	0,87%
50	4	597,4	597,4	599,52	0,35%	597,4	602,88	0,92%
50	4	638,4	638,8	643,2	0,75%	638,4	641,56	0,49%
50	4	623,2	626	629,72	1,05%	623,2	629,76	1,05%
50	4	593,2	596	602,36	1,54%	593,2	597,64	0,75%
50	4	579,4	581,2	583,32	0,68%	579,4	584,96	0,96%
50	4	580,4	586,2	590,64	1,76%	580,4	585,52	0,88%
50	4	437	437	442,28	1,21%	439	441,96	1,14%
50	4	517	517	520,48	0,67%	517	519,84	0,55%
50	4	525,4	532,6	535,84	1,99%	525,4	533	1,45%
50	4	599,2	599,2	603,72	0,75%	599,8	603,04	0,64%
50	4	456,8	459,6	461,72	1,08%	456,8	460,24	0,75%
50	4	516	516	518,48	0,48%	516	519,4	0,66%
50	4	493,8	494,2	497,56	0,76%	493,8	500,72	1,40%
50	4	524,2	529,6	531,76	1,44%	524,2	531,12	1,32%
50	4	528,8	528,8	534,08	1,00%	530	536,64	1,48%
50	4	531,8	531,8	534,84	0,57%	536,2	537,48	1,07%
50	4	547,8	547,8	551,52	0,68%	549,8	550,96	0,58%
50	4	575,2	575,8	577,68	0,43%	575,2	577,28	0,36%
50	4	567,8	571,4	574,2	1,13%	567,8	571,36	0,63%
50	4	525,6	525,6	530,12	0,86%	527	531,32	1,09%
50	4	558,8	558,8	564	0,93%	560,2	565,92	1,27%
50	4	613,2	613,2	618,76	0,91%	615,4	620,36	1,17%
50	4	573,4	573,4	576,32	0,51%	577,8	582,2	1,53%
50	4	540,4	544,2	548,56	1,51%	540,4	547,6	1,33%
50	4	538,2	538,2	541,76	0,66%	539,4	545,48	1,35%
50	4	467,4	468,4	473,52	1,31%	467,4	473,52	1,31%
50	4	515,6	515,6	517,16	0,30%	516	518,52	0,57%
50	4	552,2	553,4	557,68	0,99%	552,2	559,2	1,27%
50	4	505,8	507	514,92	1,80%	505,8	512,56	1,34%

Ie	0,92%	Iterations	372 296	Ie	0,94%	Iterations	363 982
Variance	0,00002	Time (s)	46,7	Variance	0,00001	Time (s)	45,5
50,3	0,93%		351 128	50,3	0,93%		352 657
	0,00002		38,0		0,00002		38,2
50,4	0,91%		393 463	50,4	0,94%		375 307
	0,00002		55,3		0,00001		52,8

			ESASI 0.8			RSASI 0.8		
n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie
50	3	878,6	884	886,68	0,92%	878,6	887,72	1,04%
50	3	924,4	924,4	934,64	1,11%	925,2	930,32	0,64%
50	3	969,6	969,6	977,28	0,79%	974	981,96	1,27%
50	3	909,8	917,8	919,68	1,09%	909,8	917,04	0,80%
50	3	883,4	883,4	888,52	0,58%	884,4	890,72	0,83%

n	m	[min] total	Min	Avarage	Ie	Min	Avarage	Ie
50	3	865,8	865,8	872,72	0,80%	874,2	879,44	1,58%
50	3	808,6	812,6	817,8	1,14%	808,6	816,12	0,93%
50	3	839,6	841,8	845,56	0,71%	839,6	845,92	0,75%
50	3	895,8	895,8	898,88	0,34%	895,8	900,6	0,54%
50	3	797	800,8	805,08	1,01%	797	804,24	0,91%
50	3	780,6	784,2	789,64	1,16%	780,6	789,96	1,20%
50	3	695,4	695,8	699,8	0,63%	695,4	698,92	0,51%
50	3	779,4	779,4	784,2	0,62%	781,4	785,72	0,81%
50	3	842,4	845,4	848,28	0,70%	842,4	848,24	0,69%
50	3	906,8	907	909,88	0,34%	906,8	911,28	0,49%
50	3	757,4	759,4	762,52	0,68%	757,4	764,56	0,95%
50	3	789,2	796	800,44	1,42%	789,2	795,4	0,79%
50	3	822,4	822,4	826	0,44%	825,4	828,32	0,72%
50	3	887,6	887,6	892,72	0,58%	889,6	892,84	0,59%
50	3	874	876,6	880,52	0,75%	874	878,6	0,53%
50	3	900,2	918,8	924,08	2,65%	900,2	916,08	1,76%
50	3	710,4	710,4	718,76	1,18%	710,6	717,04	0,93%
50	3	799,2	802,8	808,48	1,16%	799,2	806,08	0,86%
50	3	766,2	766,2	770,4	0,55%	769,8	774,16	1,04%
50	3	815,6	815,6	818,4	0,34%	816,4	818,52	0,36%
50	3	797,2	797,2	804,56	0,92%	800,4	807,8	1,33%
50	3	800,6	800,6	810,44	1,23%	806,8	813,24	1,58%
50	3	837,4	837,4	848,6	1,34%	842,8	847,2	1,17%
50	3	855,4	855,4	865,2	1,15%	855,8	859,84	0,52%
50	3	850,4	850,4	863,48	1,54%	860,4	866,56	1,90%
50	4	646,6	649,2	651,68	0,79%	646,6	649,28	0,41%
50	4	672,8	676,2	677,6	0,71%	672,8	677,4	0,68%
50	4	715,6	715,6	719,16	0,50%	718,4	721,08	0,77%
50	4	710,4	710,4	714,12	0,52%	710,6	716,44	0,85%
50	4	674	676,4	682,32	1,23%	674	680,72	1,00%
50	4	652,8	652,8	657,32	0,69%	656,8	661,84	1,38%
50	4	660,8	663,2	666,28	0,83%	660,8	664,48	0,56%
50	4	510,4	517,2	519,44	1,77%	510,4	515,92	1,08%
50	4	589,2	589,2	596,44	1,23%	592,2	596,8	1,29%
50	4	605	609,8	615,32	1,71%	605	611,2	1,02%
50	4	670,6	670,6	676,36	0,86%	677	680	1,40%
50	4	525,4	525,4	528,84	0,65%	530	531,44	1,15%
50	4	588,2	590,2	594,88	1,14%	588,2	593,4	0,88%
50	4	569	569	570,2	0,21%	570,2	572,92	0,69%
50	4	601	606,6	610,6	1,60%	601	608,04	1,17%
50	4	597,4	598	602,04	0,78%	597,4	602,04	0,78%
50	4	607,4	607,4	609,64	0,37%	607,4	610,68	0,54%
50	4	620,4	620,4	625,4	0,81%	621,4	628,72	1,34%
50	4	642,6	642,6	647,2	0,72%	644,6	647,12	0,70%
50	4	644,4	645,4	646,6	0,34%	644,4	649,52	0,79%
50	4	628,4	628,4	637,04	1,37%	633	637,64	1,47%
50	4	664,8	665	669,8	0,75%	664,8	672,64	1,18%
50	4	711,2	714,4	718,4	1,01%	711,2	715,56	0,61%
50	4	665,4	665,4	671,6	0,93%	670,4	673,88	1,27%
50	4	642,4	643,6	647,08	0,73%	642,4	649,6	1,12%
50	4	632,6	634,6	636,6	0,63%	632,6	639,4	1,07%
50	4	579,8	581,4	586,56	1,17%	579,8	584,84	0,87%
50	4	604,4	610,4	613,6	1,52%	604,4	611,48	1,17%
50	4	652,2	653,4	660,4	1,26%	652,2	655,44	0,50%
50	4	592	592	595,08	0,52%	592,2	595,24	0,55%

LIST OF FIGURES

All figures were elaborated by the author.

LIST OF TABLES

All tables were elaborated by the author.