UNIVERSITE LIBRE DE BRUXELLES

Faculté des Sciences Appliquées

Année Académique 2006-2007

ANT COLONY OPTIMIZATION ALGORITHM FOR BIOBJECTIVE PERMUTATION FLOWSHOP PROBLEMS

Directeur de Mémoire:
Prof. Ph. VINCKE
Cosuperviseurs:
M. BIRATTARI
Y. DE SMET
T. STÜTZLE

Mémoire de fin d'études
présenté par Trung Truc HUYNH
en vue de l'obtention du grade
d'Ingénieur civil en
Electromécanique

Abstract

Ant Colony Optimization (ACO) is a metaheuristic for solving combinatorial optimization problems which belongs to swarm intelligence approaches. It is the most successful and the most studied technique among these methods which are inspired by social behaviour of insects and other animals.

In this master thesis, we proposed two ACO algorithms using respectively one and two pheromone matrices to tackle a biobjective permutation flowshop scheduling problem where the makespan and the total tardiness are the objectives to minimize. The two algorithms are aggregation methods and their underlying idea is to force each colony to search for solutions in different directions of the space by changing the importance of each objective all along the procedure.

These two algorithms and their different variants have been studied and tested on four instances. They have been compared through the analysis of more than one hundred pairwise comparisons of the different configurations. The results obtained have helped us to have a better understanding of the problem and provide indications and suggestions for further research.

Résumé

Ant Colony Optimization (ACO) ou l'Optimisation par Colonies de Fourmis (OCF) est une métaheuristique destinée à la résolution de problème d'optimisation combinatoire. Cette technique est la plus efficace et la plus étudiée parmi les méthodes dérivées de l'intelligence en essaim qui s'inspirent du comportement social de certains insectes et de certains animaux.

Dans ce Mémoire de fin d'études, nous proposons deux algorithmes basés sur l'OCF utilisant respectivement une et deux matrices de phéromone, pour la résolution d'un problème biobjectif d'ordonnancement flowshop où le makespan et le total tardiness sont les deux critères à optimiser.

Ces deux algorithmes sont des approches agrégatives dont l'idée est de forcer chaque colonie à chercher les solutions dans différentes directions de l'espace en faisant varier l'importance de chaque objectif tout au long de la procédure.

Ces deux algorithmes et leurs différentes variantes ont été étudiés et testés sur quatre instances et comparées par l'analyse de plus de 100 comparaisons par paires de ces différentes configurations. Les résultats obtenus nous ont aidés à avoir une meilleure compréhension du problème et fournissent des indications et des suggestions pour de futures recherches plus approfondies.

Acknowledgements

I wish to thank Prof. Philippe Vincke the research Director of the operational research laboratory of the Engineering Faculty of the Université Libre de Bruxelles (SMG) for the opportunity to work in combinatorial optimization problems.

I wish to thank Yves Desmet researcher of the operational research laboratory of the Engineering Faculty of the Université Libre de Bruxelles (SMG) for his advices in writing and his support. I'm grateful to him for the opportunity to work in such an interesting field and for having been comprehensible with my special situation these last two years.

I am deeply grateful to Mauro Birattari and Thomas Stützle, researchers of the Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificelle (IRIDIA), for their helpful advices and direction throughout the entire work. I want to express my gratitude to them for having explained things clearly, helping me in the programmation and for having been available for me and all my questions.

I also wish to thank Manuel López Ibáñez for having kindly provided his script to plot the difference of two attainment functions and for his help to understand it.

I wish to express my gratitude to my parents and family, which have been understanding and supportive, contributing through their encouragement to the completion of the work.

I also wish to express my gratitude to my friends and the different physiotherapists and neuropsychologists with whom I have worked these last two years and half for their support and their encouragement.

Table of contents

List of Figures

List of tables

1

Introduction

Ant Colony Optimization (ACO) is a metaheuristic for solving combinatorial optimization problems. This population based approach was inspired by the behaviour of ants in finding the shortest path from their colony to the food.

The particularity of an ant colony is that without any explicit centralized control or any direct communication, simple agents are capable to create local interactions which lead to the emergence of a global behaviour which serves the interests of the whole population. This is the concept of swarm intelligence that can be observed in ants' colony but also in birds flocking or in bacterial growth.

The ants' system of communication is based on the modification of their environment. While searching for food, varied quantities of pheromone are laid down on the path taken by each ant and these quantities indicate the distance and the quality of the source food. Thus paths with more pheromone will be more attractive for the ants and will be preferred. This idea was at the origin of the first ACO algorithm proposed y Dorigo in 1992 [48].

Since this first work, many variants of the basic principle have been developed and applied to a variety of classical hard combinatorial optimization problems such as the travelling salesman problem [48,52,55,146], the quadratic assignment problem [67,101,147], the sequential ordering problem [66], and many other applications. Very good results, sometimes state of the art were obtained for some applications, what explains that ACO is nowadays applied by many researchers to solve classical hard combinatorial optimization, dynamic or multiobjective problems. This work deals with this last field.

In this master thesis, we will propose two ant colony optimisation algorithms for a biobjective permutation flowshop scheduling problem.

Flowshop scheduling problem is frequently studied. Since the introduction of the first flowshop problem by Johnson in 1954 [87], many variants of this problem with different objectives have been tackled by many different algorithms. Although a single objective is deemed as insufficient for most real applications, most of works on flowshop problems are using a single objective approach.

In this work we will present two multiobjective approaches for the problem. We will study and compare these two approaches and their different variants in order to choose most adapted approach following the problem and the preferences on the objectives.

The remainder of this Master thesis is structured as follows:

In section 2, after having reviewed scheduling problems and common objectives to optimize in scheduling problems, we present more precisely the permutation flowshop scheduling problem and the two objectives which constitute the biobjective problem tackled in this work.

In section 3, we review metaheuristics which have been applied to flowshop problems and we present in details ACO and some of its applications.

In section 4 we present multiobjective optimization, the different existing approaches and the two ACO approaches we proposed to tackle the biobjective problem.

Section 5 deals with the experimental part of this work where results are presented and discussed.

Finally in Section 6, some conclusions and indications for further research are given.

2

Definition of the problem

In many manufacturing and production systems, different jobs have to be processed by several machines in a given order. This multi-operation situation is often reflected in a shop scheduling model, where a number of jobs are to be processed in a shop consisting of several machines. In real world thousands of possible configurations exist for the production. In this section, we will first present a classification of existing shops problems. Then the different shops problems and the different objectives which can be tackled will be described.

2.1 Classification

In a scheduling problem, many of parameters have to be taken into account, the kind of scheduling problem, the objective function, the constraints, … To make it clearer, Graham et al. have introduced in the end of the seventies a three-field notation $\alpha|\beta|\gamma$ which helps to classify the different scheduling problems, and allows to have a quick view of the kind of problem to deal with [74]. This notation may be sketched as follows:

The first field, α, indicates the machine environment. For instance, α = F or α = J denotes the flow shop or job shop model respectively. The number of machines m is either part of the problem instance or equal to a fixed constant. In the latter case, the letter m or a positive integer is added after the machine environment, i.e. a two machine job shop model is specified by $J2$.

The second field, β, consists of the job characteristics, i.e. the processing restrictions and constraints. By contrast to the first field, this field can be empty, which implies the default of non-preemptive and independent jobs. Examples of possible entries in this field are $\beta = pmtn$, meaning that preemption is allowed (i.e. the processing of any operation may be interrupted and resumed at a later time), and $\beta = prec$, meaning that there are precedence constraints between the jobs (i.e. the processing of a job cannot start before the completion of another job).

The third field γ specifies the objective to deal with. An optimality objective assesses the relative merits or performances of competing feasible schedules. Examples of commonly used criteria are minimizing the makespan $C_{\max}$ (the completion time of all the jobs on all the machines) or minimizing the total weighted mean flow time, (the average time the jobs remain in the machines). With this three-field notation it becomes possible to classify and to have a quick reference for all the variations of scheduling problems. For instance, the problem of minimizing makespan in a m-machine permutation flowshop is identified by the three-tuple $Fm \mid prmu \mid C_{\max}$, while the problem in a general flowshop problem (without permutation) m machine flow shop is denoted by $F \mid m \mid C_{\max}$.

In theory, it is possible for flowshop problems to enumerate all the $n!$ possible solutions and try them to find the best one according to an objective. This approach works for small problems but with an increasing number of jobs and machines, this method becomes too heavy for today high speed computers. A NP-complete problem is a problem which cannot be solved in a polynomial number of steps of the input size. If it can, the problem is said P-complete. Garey has shown in 1979 that flowshop scheduling problem is NP-complete [71] like other famous problems, the Travel Salesman Problem (TSP) or the Quadratic assignment Problem (QAP) for which no polynomial time algorithm is known. Thus we know that no algorithm can solve a large sized problem in a polynomial number of steps. Hence much of the efforts of researchers were intended to develop heuristics that are likely to give not necessarily optimal solutions, but good solutions. This will be developed in the section 3.

The order in which a job passes through the machines is the sequence, also called the processing route and it is fixed for each job. This processing route is one of the elements that distinguish the three typical models that can be found in the reality and in the literature:

- Open shop: there are no constraints on the machine sequence. The jobs can visit the machine in any order.
- Job shop: the machine sequences can be different for each job.

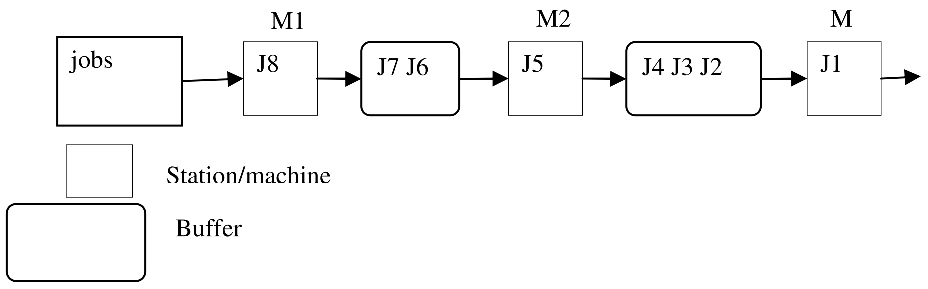


Figure 1: Illustration of a flowshop production line with buffer

– Flow shop: the most constrained shop production, all the jobs have to visit the different machines in the same order.

The job shop production will be simply presented whereas the flow shop scheduling problem will be analyzed more in details. But first we will provide in the next section a brief outline of the classical scheduling models. We make effort to adhere to traditional notation and standard terminology.

2.2 Nomenclature of parameters

Here are presented the notations used in the rest of the work:

Task t: Non-divisible activity which has to be performed in a station.

Job i: part of a subassembly or assembly, processed by a station. In a mixed model production line the jobs belongs to different models which include different processing times depending on the model at the stations.

Station j: One or more tasks may be assigned to station j. In the classical flowshop problem m stations are aligned in series and all jobs have to visit the stations in the same order. The length of station j is L_j.

Operation: The processing of a job in a machine. The operation of a job i in a machine j is characterized by its processing time p_{ij}. If a job i has to be processed on machine j the job i can start on machine j only if it is completed on machine $j-1$ and if machine j is free.

Buffer: Buffers were originally introduced between two consecutive stations to decouple them in order to avoid blocking and starving. Buffers are often located before. Figure 1 gives an illustration of a flowshop production line with buffer.

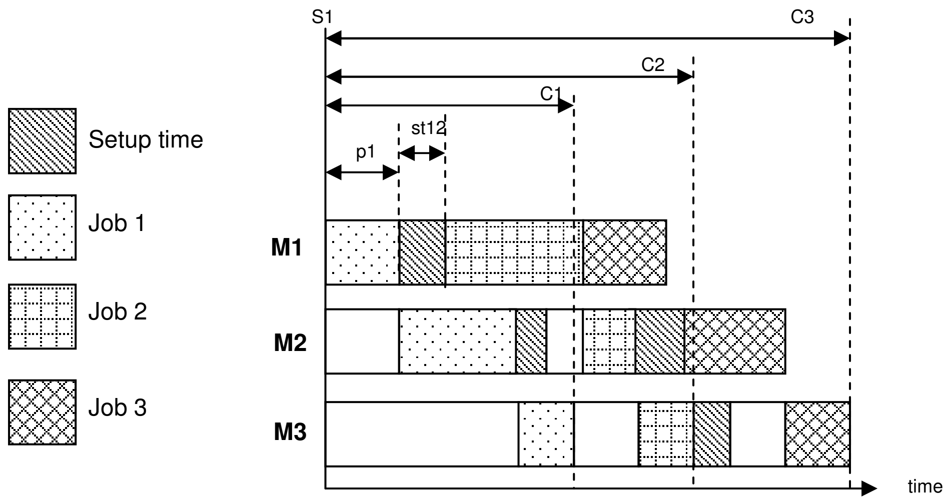


Figure 2: Illustration of some element of the nomenclature

and after bottleneck stations. The reason is that this already critical part of the production usually is the limiting section. In automobile productions buffers of enormous dimensions can be found, which in principle decouple the main successive production sections. This buffer is, furthermore, used to reorder the jobs, available in the buffer, on a large scale. In Figure 2 we present an example of a scheduling of three jobs which have to visit three machines.

Processing time p_{ij}: Also called assembly-time, is the time that a job i is maintained at station j while being processed. Due to the nature of a flowshop, a job that is not processed at a station has to pass this station with a processing time equal to zero.

Preemptive/Non preemptive: Preemptive operation means that the process may be interrupted and resumed later, even on another station. Furthermore an operation may be interrupted several times. If preemption is not allowed, the operation is called non preemptive.

Setup time st_{fgi}: Setup time is concerned if an additional time appears to change the setup of station j, in order to be able to process job $i+1$ which is of model g after a job i which is of model f. If the setup time is independent of the model, it can be simply added to the processing time.

Start-time S_i: The time job i enters the system is called start-time.

Completion-time C_i: The time job i exits the system is called completion time and is the completion time of the last job i on the last machine.

Launch-interval λ: The time between two consecutive jobs entering the production line is called launch-interval. Usually it is a constant value, also called cycle time. A constant launch-interval results in a fixed production rate (production quantity per unit of time).

Setup cost sc_{fgi}: In a similar way, setup cost is concerned if an additional cost appears to change the setup of station j, in order to be able to process job $i+1$ which is of model g after job i which is of model f. If the setup cost is independent of the model, it can be simply added to the processing cost.

Demand D: The demand describes the total volume of jobs to be processed .Usually in addition to the volume; the start-date and due-date d_i are also given. These values describe the earliest possible point of time to start working on a particular job and the date the finished product has to be delivered to the customer. Very often, penalty for delivering too early or too late are applied.

Model M: In the mixed model flowshop different models are based on the same basic product. The difference from one model to another may be due to an option that is not applied to all models or a variation of an option.

Job sequence π_j: The job sequence defines the order of jobs at station j.

A job sequence that is the same for all stations is called a permutation sequence. In flowshop, the stations sequence, the order in which the individual jobs visit the stations, is the same for all jobs.

Precedence: The precedence gives a dependency of jobs in respect to the processing. A job i is said to be predecessor of job k if job i has to be processed before job k. An immediate predecessor then is a job that has to be processed immediately before another job. In assembly, this is something very common.

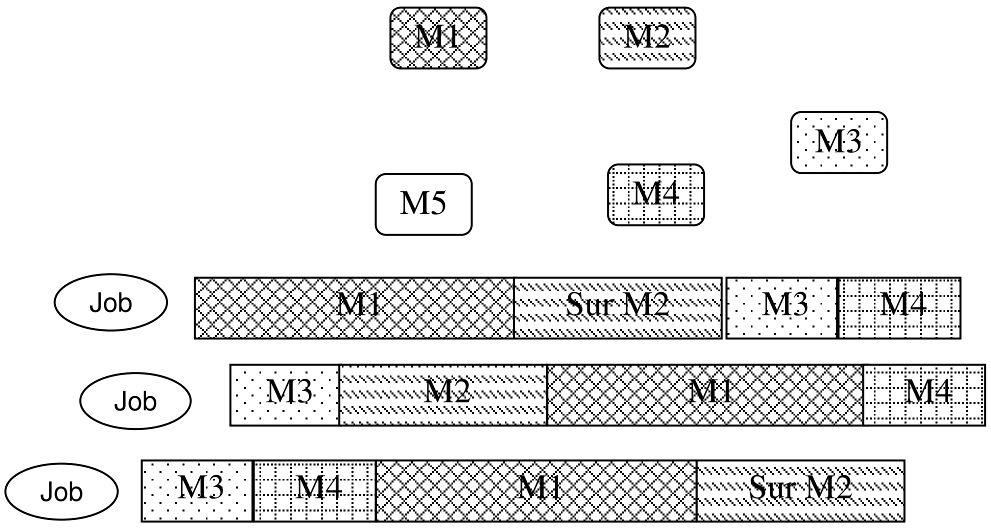


Figure 3: Scheme of an open shop production line

2.3 Open shop

. In the open shop model there is no restriction and no constraints on the movement of the jobs in the production installations. Each job can have its own machine sequence and there is no linear path between the machines. At the output of a machine, a job can go to any other machine of the production line. The Figure 3 represents the sequence of three jobs which have to be processed on five machines. In open shop, no constraint exists, each job can follow its own path between the machines.

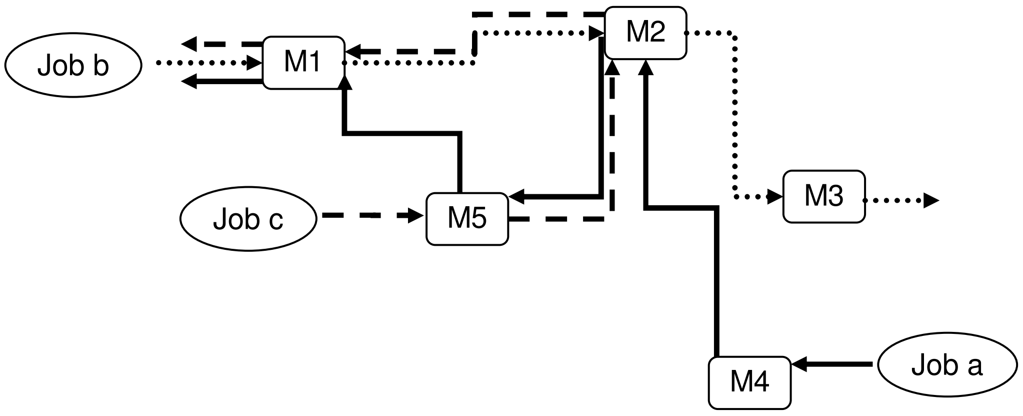


Figure 4: Scheme of a job shop production line

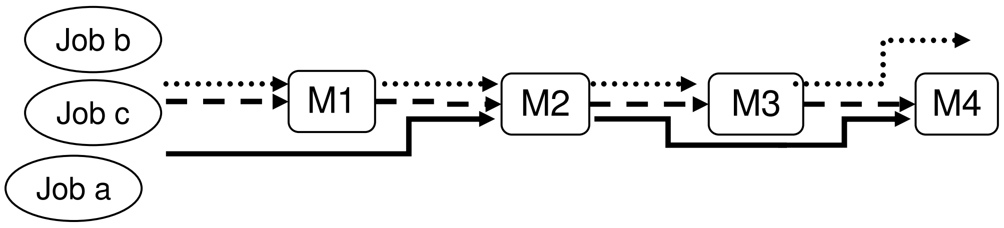


Figure 5: Scheme of a flow shop production line

2.4 Job shop

The job shop model is one of the most general models in scheduling theory. In a job shop scheduling problem, each job consists of a number of operations to be processed on all or some of the machines, and each job has its own processing routes to follow. Hence to construct a feasible schedule for a job shop, we have to determine, for each machine, the order in which the jobs have to be processed. An example of a five machine job shop installation is given in the Figure 4.

2. 5 Flow shop

The flow shop is a particular type of job shop. In flow shop, each job requires processing on every machine only once and the processing route is identical for all jobs, they all have to go through the machines in the same order. An example of a four machine flowshop production where three jobs are processed is illustrated Figure 5.

	Job shop	Flow shop
characteristics	– Equipment and staff grouped based on function – High variety - low volume – Each output processed differently	– Heavily automated special purpose equipment – High volume - low variety – Both services and products can use flow shop form of processing
advantages	– Flexibility to respond to individual demands – Less expensive general equipment – Easier maintenance and installation of general equipment – General equipment easier to modify – Dangerous activities can be segregated from other operations – Higher skilled work leading to pride of workmanship – Concentration of experience and expertise – Pace of work not dictated by moving line – Less vulnerable to equipment breakdowns	– Low unit cost – low material handling costs – low direct labour cost – – specialized high volume equipment – bulk purchasing – lower labour rates – low in-process inventories – simplified managerial control

Table 1 : Comparison job shop/flow shop production

Some characteristics, advantages and disadvantages are presented in Table 1 to present the main differences between job shop and flow shop production. Advantages of each kind of production are also presented.

In the industry, due to economical reasons a production line structured as a flowshop is something very common. In such installations, all the jobs have to go through one production line where they are processed under many machines. Initially, for flowshop production all products and jobs were the same and the processing time of jobs on each machine was the same and there was no problem of scheduling as the order in which jobs must enter the machines had no importance.

Nowadays, mixed-model flowshop production is often used and such type of production line is found in an increasing number of production environments. This is the logical result of the increased necessity of customer orientated product spectrums. In mixed-model flowshop, even if the jobs are very similar and based on the same model, they all have something specific. Hence the time each job is maintained in a machine while processing is different and then finding an optimal sequence considering one objective function has become a crucial problem for managers. Scheduling flowshop problems appear only when variation of the same product are produced on one production line.

The flowshop scheduling problem consists of finding an optimal sequence of n jobs $(J_1, J_2, ..., J_n)$ which have to visit m machines. Each job has a set of m operations, one operation per machine. π_j describes the order in which the job i has to visit the machine j. In the particular case of the permutation flowshop problem, all the jobs have to go through the machine in the same order, hence $\pi_1 = \pi_2 = ... = \pi_i = ... = \pi_n$. Furthermore, the processing time p_{ij} of job i on the machine j is known and stays constant. In theory, parameters such as setup time and setup cost have to be taken into account but because of the additional complexity, most algorithms do not consider them.

2.6 Objective functions

Different objectives depending on what is important for the production can be associated to a flowshop problem. In the following section a non exhaustive list of objectives that can be associated to flowshop problems will be given.

Objective functions can be divided in two categories: time orientated and cost orientated.

2.6.1 Time orientated functions

Makespan, $C_{\max}$**,** is the most common objective used in scheduling problem. The objective is to minimize the maximum completion time necessary to process all the n jobs on m machines. It is also called the total production time. Minimizing the makespan normally ensures a high utilization of the production resources and early satisfaction of the client demand.

Makespan is defined as:

$$\max\{C_i | \, i = 1..., n\}$$

Maximum flow time, $F_{\max}$ is to minimize the flow time. The flow time is the period of time between the beginning and the end of a job; it is also the time the job stays in the production line. It would be noticed that if all the released dates of the jobs are equal to zero, the Maximum flow time is equal to the makespan.

Maximum flow time is defined as:

$\max\{(C_i - S_i) \mid i = 1..., n\}$ where S_i is the release date of the job i.

A weight ω_i can also be associated to each job. In this case the purpose is to minimize the weighted flow time.

Weighted flow time is defined as:

$$\sum_{i=1}^{n} \omega_i (C_i - S_i)$$

Minimization of this variable leads to stable utilization of the resources, rapid turn-around of jobs and the minimization of the in-process inventory. Minimizing $F_{\max}$ leads also to the minimization of the works in process (WIP) which are a very important factor of cost in production.

Mean flow time $\overline{F}$ represents the average time the jobs remain in the machines. Minimizing this variable also leads to the minimization of the WIP.

Mean flow time is defined as:

$$\sum_{i=1}^{n} (C_i - S_i)/n$$

Weighted mean flow time is defined as:

$$\sum_{i=1}^{n} \omega_i (C_i - S_i)/n$$

Setup time may occur in a mixed model production, setup time sc_{fgi} when at station j a job $i+1$ of model type g follows job i of model type f. Minimizing total setup time, furthermore, ends to decrease the total flow time.

Setup time is defined as:

$$\sum_{i=1}^{n} st_{fgi}$$

Idle time I_{ij} is the amount of time that a job is waiting to be processed at a machine (because the machine is unavailable), after it has been processed by the previous machine.

Idle time I_{ij} at station j occurs when an operator is kept waiting for job i. This may be caused by a job that has not yet arrived, or because an auxiliary operator is still occupied with the job. When it occurs that setup time is separable from the processing time, the operator can benefit from this idle time in order to perform the necessary changes for the next job to be processed. Minimizing the total idle time leads to the minimization of the time that a work station is not producing.

The Idle time is defined as:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} I_{ij}$$

The mean Idle time is defined as:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} I_{ij} / m$$

Utility time U_{ij} is the time an auxiliary operator is required to help an operator who has to work on a job $i+1$ before having finished with job i.

The Utility time is defined as:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} U_{ij}$$

Mean utility time is defined as:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} U_{ij} / m$$

Total tardiness, T_i is the difference between the completion time and the due date d_i of the job i, considering that the job is completed after its due date. A weight ω_i is associated to each job and the objective is to minimize the total weighted tardiness.

Tardiness T_i is defined as:

$$\max\{(C_i - d_i) \; | i = 1,...,n\}$$

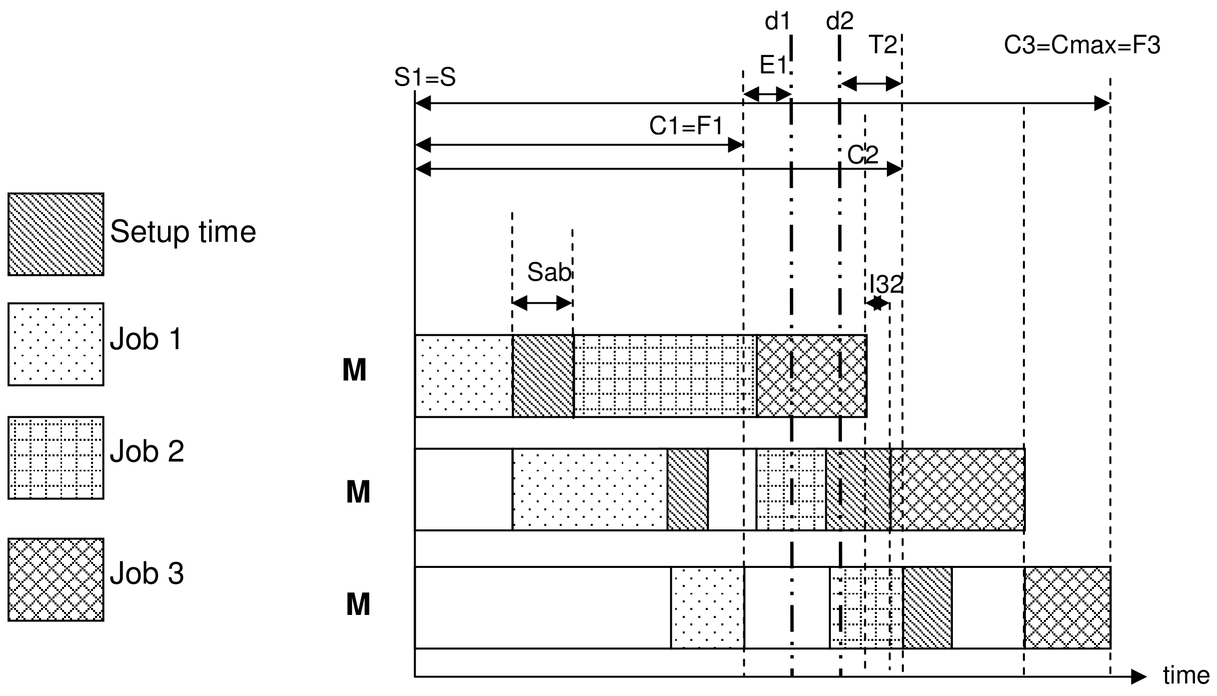


Figure 6: Illustration of some objectives

Total tardiness is defined as:

$$\sum_{i=1}^{n} T_i$$

Total earliness: is the same problem as the total tardiness except that in this case, penalty occurs if the jobs are completed before the due dates. Minimizing the total weighted earliness also means minimizing the costs due to the obligation of stocking the finished jobs.

Earliness E_i is defined as:

$$\max\{(d_i - C_i) \; | i = 1,...,n\}$$

The earliness and the tardiness criteria can be combined and the objective becomes the minimization of the sum of the earliness and the tardiness.

In the Figure 6 you will find a graphical representation of these different values with job *1* of model *a* and job *2* of model *b*.

2.6.2 Cost orientated functions

Setup cost: The occurrence of setup cost in a production may lead to the objective of minimizing the total setup cost to keep the production costs reasonable.

Setup cost is defined as:

$$\sum_{i=1}^{n} sc_{fgi}$$

2.7 Diversity of flowshop problems

Next to the classical flowshop problem, several other variations of this problem exist. In practice, lots of different industries used flowshop productions; each industry often has its own needs and its own specificities. For example in chemistry it is common practice that once a job is started, it cannot be interrupted, which leads to a non wait flowshop problem. We will now present different types of flowshop problems which are studied in literature.

Non permutation flowshop: The first to mention the flowshop problem was Johnson in 1954[87]. In this problem, n jobs have to be processed on m machines arranged in series according to the sequence of the operations. Here are the rules of this problem:

- each job has the same machine sequence; they all have to visit the m machines in the same order.
- each job can be processed only on one machine at one time and each job is processed only once on each machine.
- each machine can process only one job at the same time.
- jobs may bypass another job between two machines.

The problem consists in finding the best job sequence for each work station j according to one objective function.

Permutation flowshop: here the solutions are limited. One job cannot bypass another between two machines hence the job sequence on the first machine is maintained for all the other machines in the production line and solutions are limited to job sequences, π_i with $\pi_1 = \pi_2 = ... = \pi_i = ... = \pi_m$. It is typically the case in no-wait flowshop problem or for some assembly.

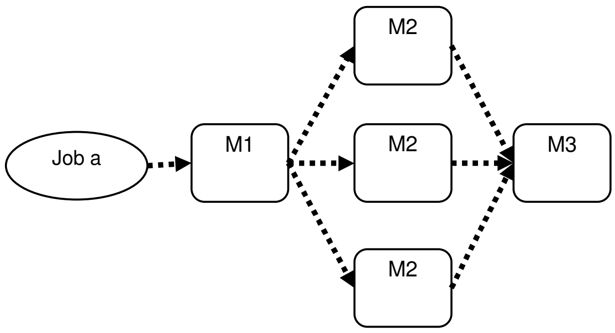


Figure 7: Illustration of identical parallel stations flowshop

Zero-buffer and **no-wait** flowshop: in these two variations, jobs are not authorized to form queues between two machines. With a buffer capacity equal to zero when a job i is finished on a machine j, it can move to the next machine $j+1$ only if the machine $j+1$ is free, only if there is no job processing on this machine. In the no-wait flowshop problem it is more restrictive. When a job i has begun its processing on the first machine, it must continue without any delay to be processed on each of the m machines.

Hence the only sequences authorized are the ones which do not lead to the blocking of any machines. Practical applications of this problem can be found in the chemical and pharmaceutical industry, in the service industry, and in the metal industry.

No-idle flowshop: In this problem, when a machine has started processing, it must do all the operations assigned without any interruption. As shown by Cepek and al. in 2002 [23]**,** this case can appear in real life when a company needs to rent expensive equipment for the duration of the operation. Hence solving a no-idle flowshop problem permits to minimize the renting time of the expensive equipment.

Flexible-Hybrid-Compound flowshop: in this case, parallel stations exist. Parallel stations reduce cycle time needed for an operation on a station. In the mixed-model case the processing time of a job on a machine depends on its model, thus the parallel station gives the opportunity to one job to overtake its predecessor. In this type of problem two kinds of setup exist: identical parallel stations and non-identical parallel stations (see Figure 7 and 8). Identical parallel stations accelerate the process while non identical parallel stations allow a job to overtake another one.

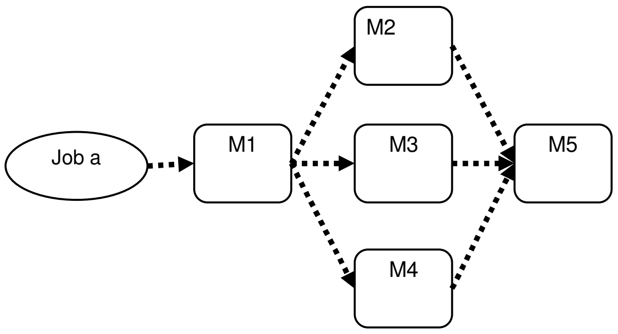


Figure 8: Illustration of non-identical parallel stations flowshop

2.8 Biobjective permutation flowshop problem

In this work, we will focus on a biobjective permutation flowshop problem.

There are several assumptions that are commonly made regarding this problem:

- Each job i can be processed at most on one machine j at the same time.
- Each machine j can process only one job i at a time.
- No preemption is allowed, i.e. the processing of a job i on a machine j cannot be interrupted.
- All jobs are independent and are available for processing at time 0.
- The setup times of the jobs on machines are negligible and therefore can be ignored.
- The machines are continuously available.
- In-process inventory is allowed. If the next machine on the sequence needed by a job is not available, the job can wait and joins the queue at that machine.

Thus the problem is to find a permutation π_i of the n jobs which will be considered as a compromise solution between the two chosen objectives:

- the makespan
- the total tardiness

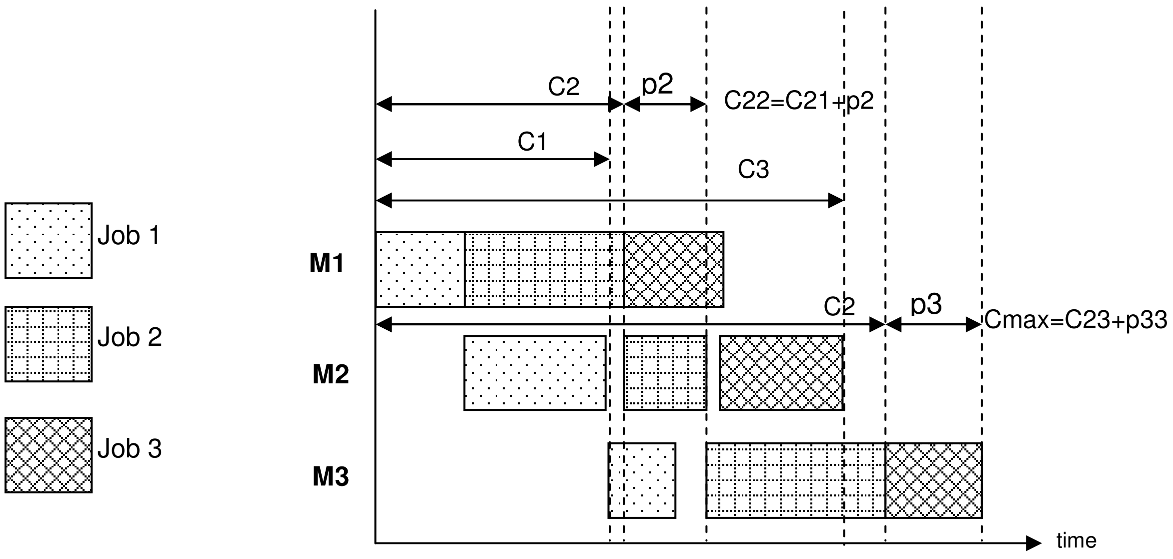


Figure 9: Illustration of the makespan

2.8.1 Makespan

Makespan objective, as said before, is the most studied objective in flowshop scheduling problem. The objective is to minimize the maximum completion time necessary to process all the n jobs on m machines. Makespan is illustrated in Figure 9.

Set C_{ij} : the completion time of job i on machine j, the makespan can be computed as follow:

$$C_{11} = p_{11}$$

$$C_{i1} = C_{(i-1)1} + p_{i1} \qquad i = 2,...,n$$

$$C_{1j} = C_{1(j-1)} + p_{1j} \qquad j = 2,...,m$$

$$C_{ij} = \max\{C_{(i-1)j}, C_{i(j-1)}\} + p_{ij} \qquad i = 2,...,n; j = 2,...,m$$

The makespan $C_{\max} = C_{nm}$, it is the completion time of the last job on the last machine.

The purpose is to finish the production as fast as possible. So in the permutation flowshop problem with the makespan criterion, the problem is to find a permutation π^* which belongs to the set Π of all the possible permutation such that $C_{\max}(\pi^* \leq C_{nm}(\pi_i)\forall\pi_i \in \Pi$.

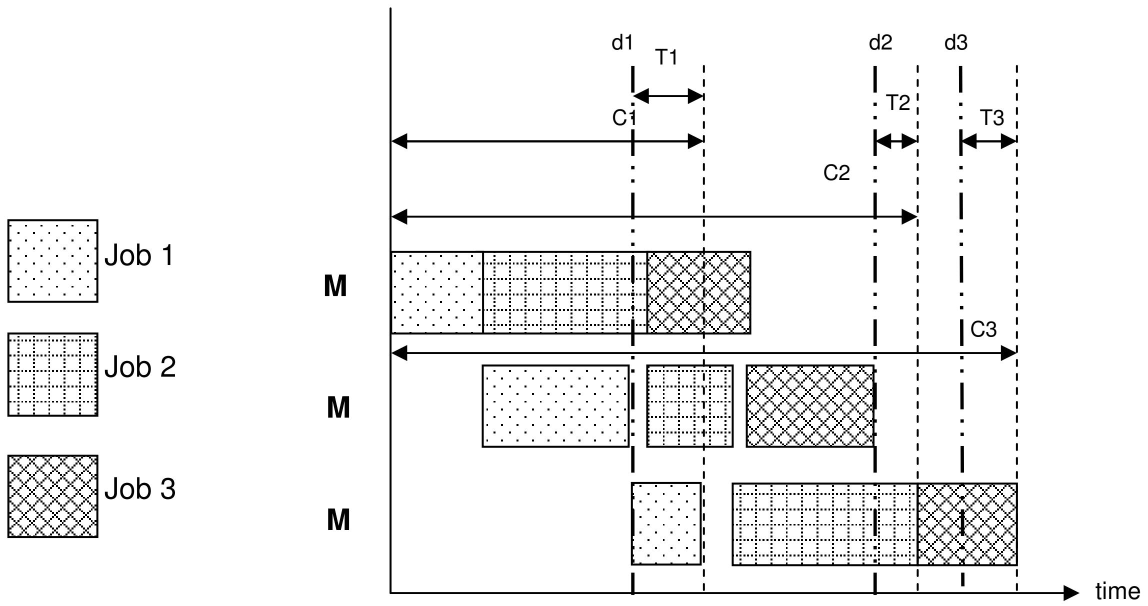


Figure 10: Illustration of the total tardiness

2.8.2 Total tardiness

In reality, tardy penalties are often associated to a job. If the products are not delivered on time, the company has to pay extra costs to compensate this delay. The objective here is to minimize the total tardiness which can also mean minimizing the tardy penalties in some cases. If $T_i = \max\{(C_i - d_i)\}$, the tardiness of this job, then

the total tardiness is:

$$\sum_{i=1}^{n} T_i$$

The notion of tardiness is illustrated in Figure 10.

If we use the Graham classification, we will be face a $Fm|Permu|C_{\max}$ associated with a $Fm|Permu|\overline{T}$.

Makespan and total tardiness are two common criteria. Furthermore, a low makespan increases machine utilization and throughput. However, the best possible makespan might sacrifice due dates and therefore both objectives are not completely correlated. In the experiments section, we will calculate the correlation of the two objectives for each instance used in the tests.

3

Optimization algorithms for Permutation Flowshop Scheduling Problem

Since the flowshop was introduced in 1954 by Johnson, lots of methods have been developed to attack this problem and all issues related such as the permutation flowshop problem. In this section, The main algorithms will be introduced in this chapter – but first both following concepts that will be used throughout need to be clarified.

- Heuristics: the purpose of heuristic algorithms is to solve a problem, not to find an optimal solution, but an approximate good solution when the time of resources are limited.
- Metaheuristics: it is a heuristic method for solving a very general class of computational problems by combining user given black-box procedures, usually heuristics themselves, in a hopefully efficient way. Metaheuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm or heuristic; or when it is not practical to implement such a method.

In the literature, it is possible to find some overview on the methods applied to the PFSP, but a global comparison between all methods cannot be found. It is very difficult to compare all the results found in the literature because of the mismatches both in the data sets and in the computer mainframe. Hence since the evaluations are partial and that there is no standard in the benchmark used, results cannot be reproduced. The different methods used for the solution of a m-machines PFSP with makespan or total tardiness for objectives are now presented. In a first time, we will present heuristics and metaheuristics used to solve flowshop scheduling problems with the makespan or the total tardiness for objective.

Finally, a third section will be used to give a detailed description of the Ant Colony Optimization (ACO) and its different applications.

3.1 Heuristics

For all the different methods, we first present existing methods for the makespan and then for the total tardiness.

3.1.1 Constructive heuristics

They are heuristics that build a feasible schedule starting from scratch.

Makespan

In 1954, Johnson developed the first known heuristic to build an optimal solution to a two machine flowshop scheduling problem. It can be used as a heuristic for a m machines problem by clustering the m machines into two "virtual" machines. Several authors like Campbell (1970) with an algorithm called CDS where it builds $m-1$ scheduling by clustering the m machines into two virtual machines and to use the Johnson algorithm to solve each 2 machines problem. Several authors like Dudek and Teuton [56] or Koulamas [95] have also developed algorithms using Johnson‘s one to solve m machines problems. It has been shown that for problems with more than three machines, the schedules are not necessarily optimal [126].

Palmer (1965) exploited another approach, namely to assign a weight or index to every job. Then the sequence is arranged by sorting the jobs according to their index [121]. He developed a heuristic consisting in calculating a "lope index" for every job, and then scheduling jobs randomly. Using this index idea, Gupta, by exploiting similarities between scheduling and sorting, proposed some modifications to Palmer's heuristic [77]. Others like Bonney and Gundry [13] or Hundal and Rajgopal [82] have used the same approach to develop their own method.

Dannenbring has introduced Rapid Access (RA) in 1977. This heuristic exploits both Johnson's algorithm and Palmer's slope index. The two virtual machine problem is defined as it is in the CDS algorithm. But the difference is that Johnson's algorithm is applied to the two weighted schemes calculated for the two machines instead of being applied to the processing time [34]. The weighting schemes give the processing times for the jobs in the two virtual machines. This algorithm provides a good solution in a short time.

All the jobs in the problem form a permutation, hence lots of methods have been proposed with the idea of exchanging the position of the jobs or inserting the jobs at different positions. In 1961, Page also used the similarities between scheduling and sorting to propose three heuristics based on sorting methods [119]. The purpose is to first obtain a good sequence and then improve it by means of jobs exchange.

The NEH heuristic is considered as the best heuristic for PFSP and has been introduced by Nawaz et al. in 1983 [114]. It is based on the idea that jobs with high processing times on all the machines should be scheduled in the sequence as early as possible. The procedure is straightforward:

1. calculate the total processing time of each job i: $P_i = \sum_{j=1}^{m} p_{ij}$
2. sort the jobs in non increasing order of P_i and take the two first one and compare the two schedules obtained by beginning by the first job and then by the second one. Choose the best solution.
3. for job $i = 3,...,n$ place the job i at each of the possible position in the sequence obtained so far and chooses the best partial schedule.

It can be noticed that Taillard has developed a speed up technique which permits to test all the partial schedules obtained by placing the job in the different positions in one single step [151].

Based on the idea of minimizing the idle time on the last machine, Sarin and Lefoka have proposed their heuristic [136]. Actually, increasing the idle time on the last machine has for consequence an increase in the makespan or in the total completion time. In this method, the sequence is completed by adding one job at a time. The job added is the one which minimizes the idle time on the last machine. If it is compared to NEH, this heuristic gives good results only when the number of machines exceeds the number of jobs.

To summarize, three main heuristic approaches exist for the PFSP:

- heuristics based on Johnson’s algorithm
- heuristics based on Palmer’s slope index
- heuristics based on insertion methods

Lots of approaches are based on one or more of these approaches but it also exists methods which are not based on these approaches. Different heuristics based on dispatching rules have been developed.

Total tardiness

For the total tardiness objective, simplest heuristics are based on dispatching rules. These rules which defined which job will be added to the sequence obtained so far. We will now present main rules used for the total tardiness problem.

Let s be the sequence of jobs that are scheduled so far, t the time at which jobs are considered for selection, $C_i(s)$ the completion time of job $i \notin s$ if it is scheduled at the end of the sequence. The different dispatching rules are:

- Earliest due date (EDD) : at time *t*, the job with minimum d_j is selected
- Earliest Weighted due date(WDD): time *t*, the job with minimum $\omega_j d_j$ is selected
- Earliest due date with Processing Times(EDDP): at time *t*, the job with minimum value $d_i \big/ \sum_{j=1}^{m} p_{ij}$ is selected
- Modified due date (Mdd): at time *t*, the job with minimum value of $\max\{d_i, C_i(s)\}$ is selected
- Slack(SLACK): at time *t*, the job with minimum value of $d_i - C_i(s)$ is selected
- Slack per Remaining Work (SRMWK): at time *t*, we select the job *i* the minimum value of $(d_i - C_i(s)) \big/ \sum_{j=1}^{m} p_{ij}$ is selected
- Shortest Processing Time(SPT): at time *t*, the job with minimum value of $\sum_{j=1}^{m} p_{ij}$ is selected
- Longest Processing Time (LPT): at time *t*, the job with maximum value of $\sum_{j=1}^{m} p_{ij}$ is selected

Four simple heuristics were proposed in [72]. They are based on dispatching rules and give priority to the job which is most expensive to hold.

In [120], Ow proposed a heuristic called Idle Time Rule (IDLE). The algorithm is based on the notion of bottleneck machine. A bottleneck machine is the machine that forces succeeding machines to be idle because it is unable to complete jobs on time.

The study deals with a proportionate flowshop where a constant of proportionality k_i is associated to each machine and thus a job has processing time $k_1 p, k_2 p, ..., k_m p$ on the respective machine. Bottleneck machine is typically a machine which has longer operation times. The characteristic of the bottleneck machine is that as soon as it is free, the job with the highest priority must be scheduled. Thus the algorithm consists in determining the bottleneck machine and scheduling the jobs with highest priorities first.

A NEH version for total tardiness also exists. In [90], jobs are sorted following the Earliest due date rule, in non decreasing order of due date or following the Latest due date rule, in decreasing order of due date. The two algorithms were called NEH_{Edd} and NEH_{Ldd}.

In the same work, they proposed another algorithm called ENS which starts from a solution constructed by EDD and improves the solution by interchanging pair of jobs.

Other methods based on algorithms for one and two machines problems have been developed. One algorithm called Modified Focused Scheduling (MFS) [120] consider *m* machines such that each machine is considered to be bottleneck. For each machine, one schedule is constructed, thus at the end of the process, the best sequence among the *m* sequences constructed is chosen. Based on the algorithm Botflow proposed for a one machine problem and described in [112], Flowshop Decomposition (FSD) uses Botflow rule to generate *m* sequences. Among these *m* sequences, the best one will be chosen for solution [2].

3.1.2 Improvement heuristics

Makespan

By contrast, improvement heuristics start with a schedule already built and try to improve this solution. In 1977, Dannenbring proposed two improvement heuristics: Rapid Access with Close Order Search (RACS) and Rapid Access with Extensive Search (RrAES) [37]. In RACS, the method consists in swapping two adjacent jobs in a sequence obtained by RA. The best schedule between the $n-1$ schedules is given as a result. This is repeatedly applied while the heuristic finds improvement.

Ho and Chang developed a method that works with the idea of minimizing the Idle time [80]. The authors refer to this time as ‘‘gap’’. After having calculated all the gaps for every pair of jobs and machines, they swap the jobs depending on the value of the gap associated to each job. This improvement heuristic has been applied after CDS heuristic of Campbell.

Suliman recently developed an improvement heuristic based on job pair exchange mechanism with a directionality constraint which reduces the size of the search space [128]. For example, if by moving a job forward, a better schedule is obtained, it is assumed that better schedules can be achieved by maintaining the forward movement and not allowing a backward movement.

Total tardiness

Kim et al. proposed in [91] two local search algorithms, ENS 1 and ENS2 which starts from a solution given by NEH_{Edd} . They use improvement procedure based on interchange and insertion of jobs.

3.2 Metaheuristics

In PFSP, metaheuristics very often starts from a solution found by a heuristic. Among all the works that can be found in the literature some more interesting will be presented in this section. These algorithms mainly deal with local search (LS), simulated annealing (SA), tabu search (TS) and genetic algorithm (GA) or hybrid methods which associate two of the different metaheuristic classes. These classes of metaheuristics will be now briefly presented. Iterated local search (ILS) [142] will also be described as it is said to be the best metaheuristic to solve PFSP with the makespan for objective according to the experiments done by Ruiz and Maroto [135] on Taillard benchmark [152].

For the different metaheuristics, we will present existing methods that have been developed to tackle the makespan problem and then the total tardiness problem.

3.2.1 Hybrid metaheuristics

After having studied the different algorithms separately, researchers have realized that it is not sufficient. A skilled combination of concepts of different metaheuristics can lead to better results.

The interaction of the different methods can take place in low level, combination of functions from different metaheuristics or high level, using a portfolio of metaheuristics to automated hybridization. The complexity and the size of the problem have an enormous influence on how the different functions should be associated. A traditional scheme that recurs in the literature is to associate an exploration method such as evolutionary algorithm with an intensification method such as local search.

3.2.2 Local search

Local search can be applied to problems where the objective is to find a solution which minimizes (or maximizes) a criterion among a number of candidate solutions. After having defined a neighbourhood relation on the search space, the local search starts from a point and iteratively moves to a neighbour solution. Typically the name of the local search indicates how the neighbour solution is chosen. If the choice of the neighbour solution is done by taking the one locally minimizing (or maximizing) the criterion, the metaheuristic takes the name of hill climbing.

The termination condition of a local search can be of different types. It can be based on a time bound, on a certain degree of optimality reached or on a number of iterations without any improvement.

Local search algorithms are typically incomplete algorithms; it means that they may stop even if the solution found is not optimal. For example, it can happen that the algorithm is unable to improve its current solution as the optimal solution can lie very far away from the neighbourhood of the last best solution found.

Local search algorithms are applied in lots of different problems like the TSP, the vertex cover problem and also the PFSP. Three different local search algorithms often used in

In the literature, three local search algorithms are usually considered for the PFSP and other shop problems:

- transpose neighbourhood: swap two consecutive jobs at position i and position $i+1$;
- exchange neighbourhood: exchange jobs at position i and j
- insertion neighbourhood: remove the job at position i to insert it at position j.

Swap move is very fast but its solution is of low quality. Exchange move and insertion move give solutions of comparable quality, but by using the speed up technique developed by Taillard for the NEH algorithm [151] insert move works faster than exchange move. This is why insertion move is regarded as the best choice for PFSP when the makespan is the objective under consideration. Transpose neighbourhood is illustrated in Figure 11.

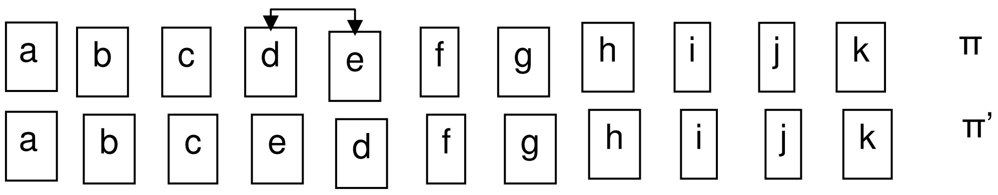


Figure 11 : Transpose neighbourhood

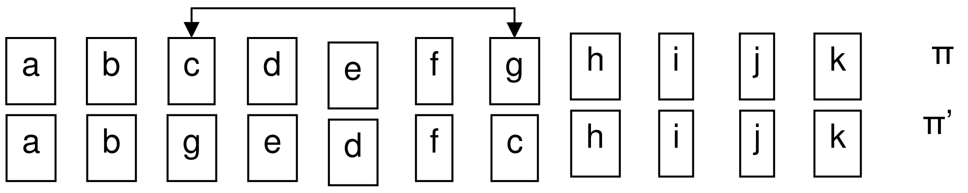


Figure 12: Exchange neighbourhood

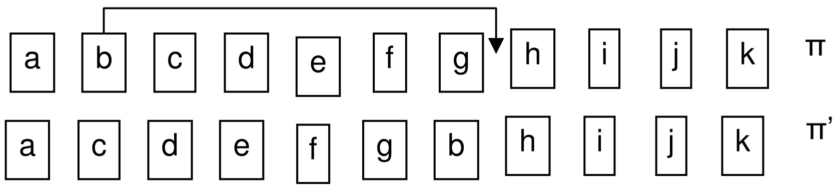


Figure 13: Insert neighbourhood

Transpose neighbourhood (see Figure 11)

In transpose neighbourhood, the size of the neighbourhood is $n-1$. In this method, a permutation π' is obtained from a starting permutation π by swapping the position of two adjacent jobs. Algorithms based on this local search give a solution quickly but do not allow to reach solutions of good quality.

Exchange neighbourhood (see Figure 12)

In this method, the size of the neighbourhood is $n \cdot (n-1)/2$. The permutation π' is obtained from the permutation π by exchanging a job at the position i with a job at position j .This local search gives good quality solutions but as the neighbourhood size is bigger, thus the exploration of the neighbourhood takes more time. Figures 12 and 13gives an illustration, of exchange neighbourhood and insert neighbourhood.

Insert neighbourhood (see Figure 13)

The size of the neighbourhood is $(n-1)^2$. The sequence π' is obtained from a sequence π by removing the job at position p_1 and by inserting it at position $p_2 \neq p_1$. The quality of the solution found is comparable to the solution given by exchange neighbourhood

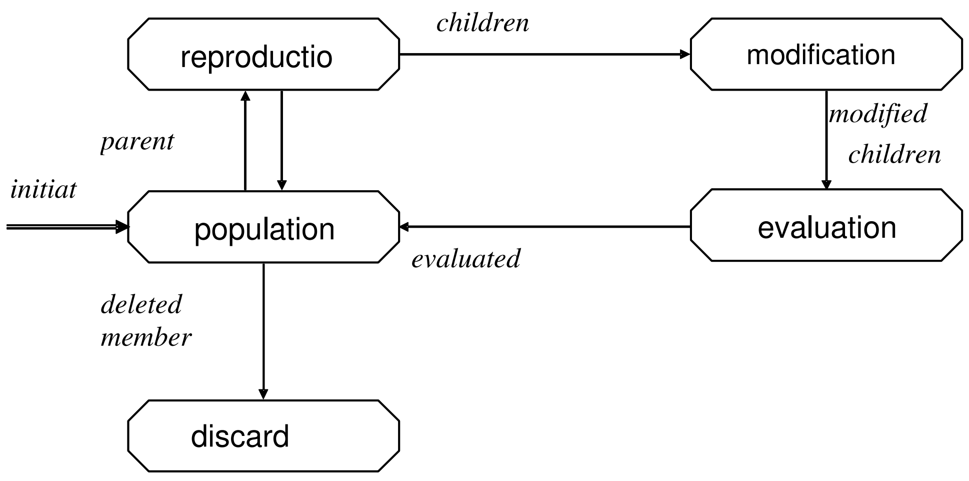


Figure 14: Cycle of reproduction in a genetic algorithm

but when the objective to deal with is the makespan, the speed up technique proposed by Taillard increases its performance and the algorithm is faster.

Nowicki and Smutnicki, in 1996 have proposed the use of specific pruning techniques that can reduce the size of the neighbourhood at each step of the search procedure and thus reduce the time needed to find a solution [115]. In the section 4 and 5, we will refer to these local search algorithms with the notations "trans" for transpose neighbourhood local search, "exchange" for exchange neighbourhood and" insert" for insert neighbourhood.

3.2.3 Genetic algorithms

This class of probabilistic optimization algorithms is inspired by biological evolution. It uses concepts of "natural selection and genetic inheritance" [35] and was originally developed by Jon Holland in 1975 [81]. In this method, a population of candidate solutions evolves toward better solutions for a given problem.

The evolution starts from a population and in each generation, the fitness of every individual is evaluated. In function of their fitness, some individuals are stochastically selected from the population. These individuals are recombined, sometimes with mutations to form a new population. This new generation will be used as a base for the next iteration. The algorithm ends when the termination condition is reached. Figure 14 shows the cycle of reproduction of a GA.

More precisely, to begin a GA procedure, genetic representations of the solution domain and a fitness function to evaluate the solution domain have to be defined. The fitness function is defined over the genetic representation and measures the quality of the represented solution. Once these elements are defined, it is possible to begin the procedure.

The first step is the initialization of the population. Traditionally several hundred or thousand possible solutions are generated randomly to constitute the first population. An important point is that this population must cover the entire range of possible solutions, all the search space

The second step is the selection. During this phase, a proportion of the existing population is selected to breed a new generation. The criterion of selection is the value of the solution according to the fitness function. The fitter solutions are more likely to be selected. Following the choice of the algorithm designer, the selection can rate all the solutions and then select the best solutions or only test a sample of the population before choosing. This function is stochastic, hence there is still a small probability that less fit solutions are chosen, what helps to keep the diversity of the operation large and helps to avoid a too fast convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection.

The reproduction phase is the third phase. Among individuals selected, the two parents are chosen and are combined by crossover and/or mutation to generate one child. This child typically shares many of the characteristics with the two parents. This operation is repeated until a new population of an appropriate size is generated. This new population, different from the first one generally have a fitness which has increased in average. GA can use two different operators to change the genes of the children:

- crossover: a genetic operator that combines the two chromosomes of the parents to produce a new chromosome which takes the best from the characteristics of each parent.
- mutation: a genetic operator that alters one or more gene values in a chromosome from its initial state.

By this procedure it is possible to construct better populations by iteration, This process is repeated until a termination condition has been reached. Common terminating conditions are:

- a solution is found that satisfies minimum criteria
- a fixed number of generations is reached

- allocated budget (computation time/money) is reached
- the highest ranking solution's fitness is reached or a plateau is found such that successive iterations no longer produce better results
- combinations of the above.

Makespan

In 1995, Chen et al. starting from the result given by CDS heuristic (Campbell) and RA heuristic (Dannenbring) to constitute the initial population, they have proposed a simple GA where only crossover operator is applied [26]. The same year, Reeves also proposed a GA algorithm starting with NEH (Nawaz) [131]. In this procedure, he uses crossover and mutation operators, but the new individuals obtained don’t replace the parents, but individuals from the generation that have a fitness value below average. The way he selects the parents is also different. Murata proposed a GA with the use of crossover and mutation operators associated with an elitist strategy [113]. But the solution obtained was first than the ones obtained by SA, TS and LS at this time, Thus he decided to implement hybrid methods associating GA and SA or GA and LS.

Total tardiness

Onwubolu and Mutingi have developed a genetic algorithm [117]. In this algorithm, they use an initial population randomly generated and combine the characteristics of the parents with crossover and mutation operators. The algorithm uses a diversity measure which guarantees a better exploration of the solution space.

3.2.4 Tabu search

Tabu search is a metaheuristic which is superimposed on another heuristic The purpose is to avoid entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited, making these moves tabu. Memory is used to store this information and the role of the memory is very important.

In a local or neighbourhood search, the algorithm starts from an initial solution and moves from neighbour to neighbour as long as possible while decreasing the value of the objective function. TS, by modifying the neighbourhood of the solution found, facilitates the exploration of the regions of the search space that would be left unexplored by the local search.

Different kinds of elements can be saved in memory. There are different kinds of tabu lists, a tabu list can contain the solutions recently visited, solutions that are excluded because they contain a specific attribute, or forbidden moves. In case some solutions are excluded because of their attributes, during the exploration of the search space, there is a risk of missing solutions of good quality which are accessible only by passing on the solutions excluded. To overcome this problem, aspiration criteria are introduced which allows overriding the tabu state of a solution to include it in the allowed set.

The other characteristic of TS is that the new courses are not chosen randomly, Tabu search proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This insures new regions of a problems solution space will be investigated in with the goal of avoiding local minima and ultimately finding the desired solution.

Makespan

For the PFSP, Widmer and Hertz introduced an algorithm constituted of two phases called SPIRIT [154]. In the first phase, a problem is generated with an analogy with the Open Travelling Salesman Problem (OTSP) and is solved with an insertion method. In the second phase, starting with the solution obtained in the first phase, a TS algorithm combined with an exchange neighbourhood is used to improve the solution. Taillard also proposed in 1990 a similar method with an initial solution constructed by NEH combined with an exchange neighbourhood local search [151]. Reeves in 1993 improved the SPIRIT algorithm by using NEH combined with insert neighbourhood to construct the initial solution [130].

Nowicki and Smutnicki proposed a Tabu Search metaheuristic where the size of the neighbourhood is reduced what really helps to improve the results. In this method, instead of moving single job, movements are made block by block [115]. This algorithm is known as one of the best for PFSP with Iterated local search.

Moccellin and dos Santos [110] presented a hybrid Tabu Search-Simulated Annealing heuristic and showed after a comparison with simple TS and simple SA that their hybrid approach was better.

Total tardiness

Adenso-Díaz [1] has modified the TS proposed by Widmer (SPIRIT) and uses as initial solution a solution obtained by MFS (Ow), Kim in [90] also proposed its version based on SPIRIT.

Other works combining TS and SA have been proposed in the literature [1] where the TS is used to reduce the size of the neighbourhood.

In [91], Kim et al. proposed four TS based on job insertion. The tabu list can contain the objectives values obtained in the previous iterations, relative position of two adjacent jobs)

3.2.5 Simulated annealing

Simulated annealing (SA) is a probabilistic algorithm for optimization problem, invented by Kirkpatrick et al. in 1983 [92] and by Cerný in 1985 [35]. SA is a generalisation of a Monte Carlo method for examining the equations of state and frozen states of n-body systems. The name and inspiration come from the annealing process used in metallurgy. Annealing is a heat treatment that alters the micro structure of a material. After having been heated, the material is slowly cooled into a uniform structure with changes in strength and hardness properties.

With the heat, the atoms move from their initial position (a local minimum of the internal energy) and move randomly through states of higher energy. The slow cooling increases the chances of finding a configuration with lower internal energy, than the initial one. In SA algorithm, the initial solution is replaced by a nearby solution chosen following a probability rule. The energy equation for the thermodynamic system is analogous to the objective function of the combinatorial problem, and ground state is analogous to the global minimum.

In annealing if the move from the initial position to the next position causes a negative change in the internal energy, the move is accepted otherwise it is accepted with a probability depending on the difference between the corresponding function values and a parameter T called Temperature. This process is repeated sufficient times to give good sampling statistics for the current temperature. The process is then repeated for a decreased temperature until T=0. Allowing moves to states with higher energy saves the method from becoming stuck at local minima.

The major difficulty in implementation of the algorithm is that there is no obvious analogy for the temperature T with respect to a free parameter in the combinatorial problem. Furthermore, avoidance of entrainment in local minima (quenching) is dependent on the annealing schedule, the choice of initial temperature, how many iterations are performed at each temperature, and how much the temperature is decremented at each step as cooling proceeds.

Makespan

One of the first SA developed for the PFSP was done y Osman and Potts in 1989 [118]. It was a simple SA algorithm using a shift and a random neighbourhood search. One year later Ogbu and Smith proposed their SA algorithm where the initialisation is done by using the Palmer and Dannenbring heuristic [116]. In 1995, Ishibuchi introduced its SA algorithm with other characteristics that shows comparable results to SA from Osman and Potts [118].

Total tardiness

Kim in [91] also proposed four SA where moves are based on insert neighbourhood or exchange neighbourhood.

In [124] and [125], two similar SA have been designed to solve a total mean weighted tardiness. They use as initial solution a solution given by EWDD rule.

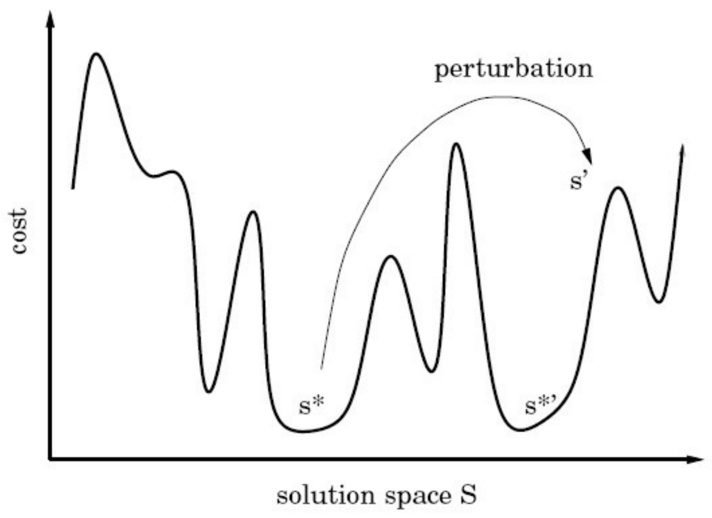


Figure15: Pictorial summary of ILS [142]

3.2.6 Iterated local search

Another metaheuristic is given by Stützle called Iterated Local Search (ILS) has been applied to the PFSP [142]. A pictorial summary is presented in Figure 15.

In ILS there are five different steps:

- generate an initial solution randomly or with a constructive heuristic
- apply a local search (s^*)
- do a perturbation: random move in higher order neighbourhood(s')
- apply a local search(s^* ')
- test with a acceptance criterion: force to lower the value of the objective function.

For the PFSP with the minimization of the makespan for objective, it NEH heuristic can be used to generate an initial solution, apply insert neighbourhood as local search, then use transpose neighbourhood as the permutation. The move would be accepted only if the makespan of the new solution is lower than the makespan of the first solution. The role of the perturbation is crucial, if it is too strong, it can be close to a random start where as if it is too weak, it can be undo by the local search. The perturbation depends strongly on the local search used and the acceptance criterion depends on both local search and perturbation used in the ILS.

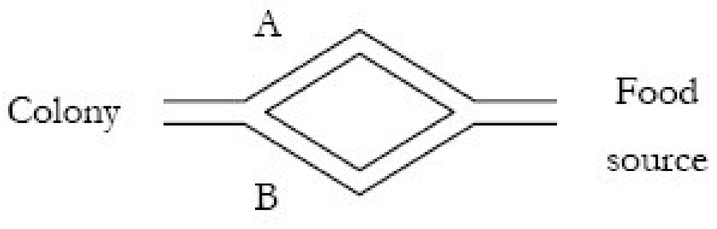


Figure 16: Illustration of the symmetrical binary bridge [40]

Ruiz and Maroto have compared different heuristics and different metaheuristics using the Taillard benchmark as instances. According to their experiments, it appears that NEH heuristic implemented with the Taillard speed up technique is the best heuristic for the Taillard benchmark. For metaheuristics, the ILS of Stützle and the GA of Reeves give better results than the other algorithms.

3.3 Ant Colony Optimization

3.3.1 The origin

The ACO algorithm has been inspired by the behaviour of real ants while they are looking for food. In real life ants initially explore the area surrounding the nest in a random manner until one ant find a source of food. It evaluates the quantity and the quality of food and carries some of it back to the nest. During its return trip, the ant deposits a pheromone on the trail on the ground. The quantity of pheromone on a route will increase the probability of an ant to choose this route, if it still finds food; it will go back to the nest following the same route reinforcing the pheromone trail. Over time, pheromone trail evaporates, thus reducing its attractive strength. Routes which are less taken will have their pheromone trail weaker as pheromone trail has more time to

evaporate. Hence the probability of taking a long route is more and more lower with time.

Actually after a while, we can remark that ants are able to find the shortest path between their nest and the food. Deneubourg et al**.** have made a famous experiment in 1990 which provide a clear demonstration of how ants self organize thanks to the pheromone trails. This experiment is called the Binary Bridge [40].

A colony of ants is separated from a food source by a bridge which divides into two branches, A and B, both of which are the same length (Figure 16). The only way out of the colony is via the bridge, so when the ants begin to explore, they are faced with the decision.

Left or right rail? Initially, there is nothing to influence the ant's decision one way or the other; it therefore has a 50% chance of choosing either branch.

Each ant lays pheromone at a constant rate as it traverses its chosen routes to and from the food source, and in the early stages of the experiment, more or less equal numbers of ants can be seen using each branch, but after some time, due to random fluctuations more pheromone are deposited on one of the branches and this in turn causes the ants to respond to the greater pheromone density, forming a preference for that branch. Finally the whole colony converges towards the use of the same branch. This positive feedback means that eventually, all ants are using the same route and there is a very low probability of any ant defecting, given the relative density of pheromone on each branch. .

Deneubourg et al. [40] propose a model of the phenomenon which matches experimental observations well. The probability p_A of an ant $i+1$ choosing either branch is calculated from the total number of ants A_i and B_i which have used each branch previously. The probability P_A that the $(i+1)$ th ant chooses branch A is:

$$p_A = \frac{(k+A_i)^m}{(k+A_i)^n(k+B_i)^n} = 1 - p_B$$

The parameter k influences the attractiveness of an unmarked trail and must be greater than zero. A low value of k will mean that small amounts of pheromone will be able to influence

ants one way or the other; a high value will ensure that a relatively large amount of pheromone has to be laid before it begins to have a noticeable influence on the ants' behaviour. The other parameter, n affects the linearity of the decision function. If n is increased, the disparity between A_i and B_i has a greater influence, causing the ant's collective decision to swing with the majority sooner, and be more difficult to reverse. Monte Carlo simulations[1](have shown that the best fit of the model is obtained for $n \approx 2$ and $k \approx 20$

[1] A Monte Carlo simulation, named after the famous gambling city, is a method of evaluating a probabilistic model using a series of random inputs.

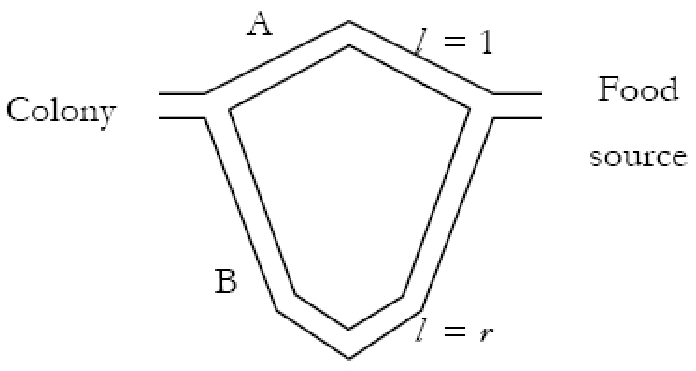


Figure 17: Illustration of the asymmetrical binary bridge [40]

The same model can be extended to situations where the branches are different in length. In this version of the experiment, trail B was longer than trail A (Figure 17), according to a ratio r. Ants using bridge B take r times as long as those using bridge A, causing r units of pheromone to be deposited on A for every one unit deposited on B. This in turn increases the likelihood of an individual ant choosing the shorter trail in future. This is called the “differential length effect”. [50] An ant choosing by chance the shorter branch will be back at its nest more quickly; therefore pheromones will be deposited on this branch more quickly than on the longer branch.

It is interesting to notice that although each ant can find a solution to bring back some food to the nest, the shortest path finding behaviour is obtained only with the whole ant colony, that thanks to the use of an indirect communication, the pheromones. We will talk about stigmergy. It is possible to talk of stigmergetic communication whenever there is an “indirect communication mediated by physical modifications of environmental states which are only locally accessible by the communicating agents [51].This is obviously the case with ants and the pheromone trail.

The idea of an ant colony algorithm is to use artificial ants to build a solution in analogy with what real ants do in their natural environment. In the ant colony optimization (ACO) metaheuristic a colony of artificial ants cooperate in constructing good solutions to difficult discrete optimization problems. The key design of this kind of algorithm is cooperation: The choice is to allocate the computational resources to a set of simple agents (artificial ants) that communicate indirectly by stigmergy thanks to the use of a parameter which acts like the natural pheromone. Good solutions are an emergent property of the agents’ cooperative interaction. Artificial ants are on one hand similar to real ants of which they are an abstraction, and on the other hand, some capabilities which are not found in real ants but which can make them more effective and efficient are added to the artificial ones.

In the following section, similarities characteristics and differences between both real and artificial ants are discussed.

3.3.2 Characteristics of artificial ants

Most of the ideas of ACO stem from real ants. In particularly on four points:

- the use of a colony of cooperating individuals,
- an (artificial) pheromone trail for local stigmergetic communication
- a sequence of local moves to find shortest paths
- a stochastic decision policy using local information and no lookahead.

We present first the similarities between artificial and real ants.

Colony of cooperating individuals.

As real ant colonies, an ACO algorithm is composed of a population, or colony, of concurrent and asynchronous entities, the ants, which cooperate to find a good "solution" to the task under consideration. Each artificial ant can build a feasible solution (as a real ant can find somehow a path between the nest and the food) but best solutions are the result of the cooperation among the individuals of the whole colony. Ants cooperate by means of the information they concurrently read/write on the problem's states they visit.

Pheromone trail and stigmergy.

Artificial ants modify some aspects of their environment as the real ants do. While real ants deposit on the path they visit a chemical substance, the pheromone, artificial ants change some numeric information locally stored in the problem's state they visit. Ant's current history and performance are taken into account by this numeric information and this information can be read and modified by any ant accessing the state. As the entities are called artificial ants, this numeric information is called artificial pheromone trail. Local pheromone trails are used as the only communication channel among the ants in ACO algorithms. This kind of stigmergetic information is very important in the utilization of collective knowledge. In fact its effect is to change the way the environment is locally perceived by the ants as a function of all the past history of the ant colony. In ACO algorithm as in the reality, pheromone trails are subject to evaporation. This mechanism modifies pheromone over time. With time, if no new pheromone are deposited, the strength of depositing pheromone decreases, which allows the ant colony to forget about its past.

Hence an ant can direct its search towards new direction without being too much constrained by the past history of the colony.

Shortest path searching and local moves.

Both artificial and real ants have the same objective, finding the shortest path (the minimum cost) joining an origin (the nest) to a destination (the food). In reality ants don't jump, they walk through adjacent terrain to find a path until the food. Artificial ants are doing as well, they are moving step-by-step through adjacent state of the problem. These terms of state and adjacency depend on the problem under consideration.

Stochastic and myopic state transition policy.

For both ants, artificial and real, a probabilistic decision policy is applied to go from a step to the next one while building one solution. Another point in the applied policy is that for both ants, the policy is completely local, in time and in space, there is no use of lookahead to predict future states. In fact this policy is function of the problem specification and of the local modification of the environment induced by the past of ants if we considered artificial ants and this policy is function of the terrain's structure and the modification of the environment induces by the past of the ants if we consider real ants.

But as said before artificial ants have also their specific characteristics:

- artificial ants live in a computer world, a discrete world therefore their moves consist of transitions from discrete states to discrete states.
- artificial ants have an internal state. Which allow them to keep in memory their past actions
- artificial ants can deposit an amount of pheromone which is a function of the quality of the solution
- artificial ants timing in pheromone laying is problem dependent and often does not reflect real ant's behaviour. Very often the update of the pheromone trails is done after having constructed a solution.

To improve overall system efficiency, ACO algorithms can be enriched with extra capabilities like lookahead [107], local optimization [53,47], backtracking[41,43].

The characteristics of the artificial agents now defined, the description of an ACO algorithm will be given

3.3.3 Description of the ACO metaheuristic

In ACO algorithms a finite size colony of artificial ants with the characteristics described above collectively searches for good quality solutions to the optimization problem under consideration. Each ant builds a solution, or a component of it, starting from an initial state according to some problem dependent criteria. During the building of its solution, each agent collects information on the environment, on the characteristics of the problem and also on its own performance according to the objective to optimize. The ant uses this information to modify the environment problem seen by the other ants. Ants show a cooperative behaviour, they can act concurrently and independently. As said before ants use stigmergetic communication, they use an incremental constructive approach to search and to build a feasible solution.

Each constructed solution can be expressed as a shortest path or a minimal cost in accordance with the problem constraint through the state of the problem. Ants are made such that they can always find a feasible solution which is probably poor but good quality solutions emerge as a result of the cooperation of all the ants of the colony that have all built their own solution concurrently.

According to the assigned notion of neighbourhood which is problem-dependent, to build its solution, each ant moves through a finite sequence of neighbour states. These moves depend on a stochastic local search policy directed by two different kinds of information:

- the past history of the ant stored in its memory
- Pheromone trail and a priori problem specific information, the heuristic information

The information stored in the ant’s memory can be the value or goodness of the generated solution, the contribution of each executed moves. Moreover this memory plays a fundamental role to manage the feasibility of the solution. In fact in combinatorial optimization some moves available to an ant can take them to an infeasible state. Thanks to the exploitation of ant’s memory which stored the effect of an action that can be performed in a local state) this kind of problem can be avoided.

The local, public information available for each ant comprises both some problem-specific heuristic information, and the knowledge, given by the pheromone trails, accumulated by all the ants from the beginning of the search process.

This time-global pheromone knowledge built-up by the ants represents a shared local long-term memory that influences the ants' decisions at each step of the solution building. The Characteristics of the releasing of the pheromones (when and how much pheromone should be released) depends on the problem under consideration and the design of the implementation. three possibilities exist for the time of releasing. Ants can release the pheromone while building the solution (online step-y-step, a local update) or only after an entire solution has been built (online delayed, a global update) or both.

In ACO algorithms functioning, auto catalysis plays a very important role. As said before, the more a move is chosen by the ants, the more it will receive pheromone and the more it will become interesting and desirable for the next ants. Generally the goodness of the solution built (or is building) influences on the quantity of pheromone deposited such as moves which contribute to a high quality solution are more rewarded, receive more pheromone. All these data, the locally available pheromone and the heuristic values defines ant-decision table. This probabilistic table is used by the ant's decision policy to direct their search in the most interesting regions of the search space.

A rapid drift of all the ants towards the same part of the search space is avoided by the presence of a stochastic component of the move choice decision policy and by the use of the pheromone mechanism discussed above. By playing with these two parameters, it is possible to determine the balance between the exploration of new unexplored regions of the search space and the exploitation of the accumulated knowledge. It can be good to notice that if necessary and feasible, the ants' decision policy can be enriched with problem-specific components like backtracking procedures or lookahead.

In the real world, ants after having built their solution do not die. In ACO algorithm once an ant has built its solution and has deposited its pheromone, this ant dies and is deleted from the system. Ants generation and activity, pheromone evaporation are two components active from a local perspective. Sometimes it can comprise other elements which have a global perspective, the daemon actions. As an example of daemon, we can take the case where a daemon is allowed to observe ant's behaviour and to collect useful information which it can use to deposit more additional pheromone, biasing in this way the ant search process from a non local perspective.

The concurrent and adaptive nature of the ACO algorithms make them very interesting for distributed stochastic problem where the problem representation is not stable (in terms of cost of environment problem) due to the presence of exogenous sources. Communication and transportation problems are intrinsically non stationery problems an exact model of the problem cannot be proposed very often. But because of the stigmergetic communication, ACO algorithms are not indented to problems where each state has a big sized neighbourhood.

Confronted with big sized neighbourhood, an ant has the choice between a huge numbers of possible moves among which to choose. Hence the probability of taking a good quality one is very small and thus there is very little difference between using or not pheromone trails. Below in Figure18 is shown the general procedure of an ACO algorithm.

First let's define some notations:

- A: set of routing tables
- P: set of probabilities
- M: memory of the ants
- Ω: set of constraints

```
1 procedure ACO meta-heuristic()
2 while (termination criterion not satisfied)
3 schedule  activities
4 ants-generation and activity();
5 pheromone evaporation();
6 daemon actions(); {optional}
7 end schedule activities
8 end while
9 end procedure
1 procedure ants generation and activity()
2 while (available resources)
3 schedule the creation of a new ant();
4 new active ant();
5 end while
6 end procedure
I procedure new active ant() {ant lifecycle}
2 initialize ant();
3 M = ,update ant memory();
4 while (current state ≠ target state)
5 .A = read local ant routing table();
6 P = compute transition probabilities(A,M, Ω );
7 next state = apply ant decision policy(P,Ω );
8 move to next state(next state);
if (online step-by-step pheromone update)
9 deposit pheromone on the visited arc();
10 update ant routing table();
11 M = update internal state();
12 end while
if (online delayed pheromone update)
13 for each visited arc ∈ ψ  do
14 deposit pheromone-on the visited arc();
15 update ant routing table();
16 end foreach
17 die();
18 end procedure
```

Figure 18: Global procedure of an Ant colony metaheuristic [63]

3.3.4 Important Choices in the application of an ACO algorithm

To apply an ACO algorithm to a new application, lots of choices have to be made and this carefully to obtain good results. The first choice to do is to define the meaning of the pheromone trail, and then the balance between exploration of the search space and exploitation of a solution has to be done. Other choices like the use of a local search, the presence of heuristic information or of a candidate list, the number of ants used in the algorithm have also to be made. All these choices are very important and have a large impact on the final results and on the efficiency of the algorithm. These different choices will be now briefly discussed.

Pheromone trails definition

When ACO is applied for a new application, the really important point is to define the meaning of the pheromone trail. In TSP, τ_{ij} can refer to the desirability of visiting a city j directly after a city i or it can also refer as the desirability of visiting city i as the j^{th} city in a tour.

In a scheduling problem, τ_{ij} can refer to the desirability of putting a job i in the j^{th} position if the objective is to minimize the makespan or the weighted total tardiness, but if the objective is to minimize the setup costs, it is better to define τ_{ij} as the desirability of putting a job j after a job i. The definition of the pheromone is crucial and a poor choice will probably lead to poor performance.

Balancing exploration and exploitation

In any metaheuristic, the balance between the two is something very important. A good exploration permits to explore unvisited regions of the search space and then increase the chance of finding a very good solution. In ACO the balance between the two can be done by several ways. First, through the pheromone trail, the pheromone trails induce a probability distribution over the search space and thus influence the regions of the search space were the solution is constructed. Depending on the distribution of the pheromone trails, the sampling distributions can be very different; it can vary from a uniform distribution to a distribution where a probability of one is assign to one solution, zero to the others, which means stagnation, . Stagnation is the situation where all ants are doing the same tour

The process of updating the pheromone is the simplest way to exploit the ants' experience. Depending on the strategy chosen, the update can be done according to the quality of the solution constructed or the best solution found during the search contributes strongly to the update (elitist strategy). In the case of the elitist strategy, the exploitation is more important than the exploration. Another possibility of tuning the exploitation/exploration is to introduce a pseudo random proportional rule during the construction of the solution. That will be shown later with the description of the Ant Colony System (ACS) algorithm [53].

If we consider a problem without the heuristic information, after some time, with the proceeding of the algorithm, the quantity of pheromone on the different "paths" will be different. This causes a shift from the initial uniform sampling of the search space to a sampling more focused on some specific regions of the search space. Thus with time, the exploration decreases. One of the problems than can appear as said before is stagnation. If this phenomenon appears too quickly in the procedure, some regions where good solutions could be found will never be visited. Different ways have been developed to avoid this situation. In ACS, there is a local update of the pheromone during the solution construction which makes the path taken by the ants less desirable to favourite the exploration of the search space. In MMAS [145] Stützle and Hoos introduce a lower limit on the pheromone trail to guarantee a minimum level of exploration and a upper limit to avoid the stagnation. A reinitialisation of the pheromone trails associated with an appropriate choice in the pheromone update can also be a good solution to explore new regions of the search space [147] is also a way of reinforcing the exploration. The last way to balance exploration and exploitation is to play on the relative importance of heuristic information and pheromone information in the construction of the solution. The more important pheromone information is, the stronger the exploitation of the search experience is.

ACO and local search

In many applications like the TSP, the QAP or the VRP, ACO algorithms give better solutions when they are combined with a local search. After having constructed a solution, a local search can be applied and this locally optimized solution can be used for the update of the pheromone trails. The two approaches are complementary, the ACO algorithms performing a coarse-grained search while the local search locally optimizes the solution produced. In [11,53,146] it has been experimentally shown that such a combination of ACO algorithm and local search gives excellent results. It can be noticed that even if the use of local search is crucial to achieve best performances in many applications, ACO algorithms can also show good performance for problems where local search algorithms cannot be applied like network routing problem or the shortest common supersequence problem[108].

Importance of heuristic information

Heuristic information of a problem is specific knowledge of this problem available a priori (in static problems) or at run time (I dynamic problems). This information combined with the pheromone information is used in the construction of the solution. Using such information can result to an important saving of computational time. When a local search is used to improve the solution in the procedure of the ACO algorithm, the importance of heuristic information is less strong than in the case of generic ACO algorithm, but it does not prevent ACO algorithm with local search of achieving good performance, also for problems where no heuristic information is available.

Number of ants

Even if one single ant is capable of generating a solution, it is often better to use a colony of m ants, $m > 1$ for an application of an ACO algorithm. For the class of geographically distributed problem, the differential length effect can appear only in presence of a colony of ants. For the combinatorial optimization problems, the use of m ants which construct r solutions could be equivalent to the situation where one ant builds $m \cdot r$ solutions, but it has been shown that it is better to use m ants, $m > 1$ in an ACO algorithm..

Candidate lists

A candidate list is a list of promising neighbourhood of the current state. Like heuristic information they are used to help the ants to explore promising regions according to an a priori information available or to information dynamically generated. Candidate lists are particularly useful in problems with large sized neighbourhood in the construction of the solution. Actually in this case, an ant has the choice between a huge number of moves when building its solution. Thus the construction can be significantly slowed down as the probability of many ants visiting the same state is very small.

Candidate lists help to strongly reduce the dimension of the search space and thus saving computational time.

Parallel implementations

Two different strategy of parallelization exist, fine-grained, where very few individuals assigned each to one processor, frequently exchange information, among the processors. By contrast, in coarse grained strategy, a larger population is assigned to single processors and information exchange is rare. In ACO, with a fine grained strategy, it has been experimentally shown that communication between the ants could be a problem. Actually, they can spend most of their time communicating to each other and updating their pheromone trails which leads to bad performance [12,18]**.**

In the contrary, coarse grained parallelization strategy has shown much more promising results for ACO. [12,18,96,109,143]. In ACO, p subcolonies are working in parallel and exchange information at certain intervals (for example each number of iterations). The information exchange can be the pheromone matrix, a group of solutions or the best solution found in each colony. Merkle and Middendorf [96] have shown that it is better to exchange the best solutions found so far and to use it to update the pheromone trails of the subcolony than exchanging complete pheromone matrices. Middendorf wit Reischle and Schmeck have also shown in [109] that the best results are obtained by limiting the information exchange to a local neighbourhood of the colonies and not to exchange the global best solutions among all the colonies. Stützle in [143] has shown that in the extreme situation where the colonies are working independently in parallel without any communication, it can still give good performances.

Algorithm	Authors	Year	References
Ant System(AS)	Dorigo et al.	1991	[48]-[55]
Elitist AS	Dorigo et al.	1992	[54]-[55]
Ant-Q	Gambardella & Dorigo	1995	[63]
Ant colony system	Dorigo & Gambardella	1996	[53]-[66]
Max-Min AS	Stützle & Hoos	1996	[146]-[147]
Rank-based AS	Bullnheimer et al.	1997	[17]-[18]-[19]
ANTS	Maniezzo	1999	[100]
BWAS	Cordón et al.	2000	[29]
Hyper-cube AS	Blum et al.	2001	[8]-[9]

Table 2: Non exhaustive list of successful ACO algorithms [49]

3.3.5 Development of different ACO algorithms

Strongly inspired by Ant System (AS), the first work on ant colony optimization [48,49], different researchers has developed several algorithm which improved the performance of the Ant System Algorithm. These algorithms can differ from AS in the construction rules of the solution or in the management of the pheromones. In Table 2, main improving algorithms are given.

We will first present Ant system [48,54] algorithm which is the base of the other algorithm before presenting some other ones. The travel salesman problem will be use as example.

An ACO for the Travelling Salesman Problem

The travelling salesman problem (TSP) was the first problem to which the algorithm has been applied to.

In this famous problem, a salesman has to visit a set of cities. The distances between the different cities are known and the goal is to find the shortest tour that allows each city to be visited once and only once by the salesman. This problem can be represented by a graph; the vertices correspond to the cities and the edges correspond to the connexion between the cities.

In Ant Colony Optimization, a number of artificial ants move in the graph. Pheromones are associated to each edge and they influence the way ants are visiting the cities. At each iteration, an ant builds a solution by walking from one vertex to another with the constraint of visiting each vertex only once. When an ant is at a vertex i it chooses the following vertex j according to a stochastic mechanism which is influenced by the pheromone. The probability p_{ij} of choosing this vertex j is proportional to the quantity of pheromone on the edge between i and j. At the end of the iteration, the values of the pheromones are modified in accordance with the quality of the solutions constructed by the ants.

Ant system (AS)

In this algorithm artificial ants build a solution by moving on the problem graph from one city to another. The algorithm executes a certain fixed number of iterations and at each iterations, m ants are building a solution by executing n steps in which a probabilistic rule is applied. At each step, an ant in the node i chooses to go to a node j through the edge (i, j) and this arc is added to the solution. The repetition of this step stops when the ant has completed its tour. In the first steps, three AS algorithms [48,54,55] were developed. They differed by the way the update of the pheromone trails is managed. Between them, two algorithms updated the pheromones while building the solution while the third one updates the pheromone trail only when the solution has been constructed.

Experiments [48,54,105] show that the third one named ant-cycle's was much better than the two others. Thus the two first algorithm were abandoned while research on AS focused on a better understanding of the characteristics of the ant-cycle's. After all ants have completed their tour, the evaporation mechanism happens just before the ants deposit their pheromone.

The amount of pheromone τ_{ij} of the arc (i,j) represented the learnt desirability of going from a node i to a node j. This amount of pheromone on the arc changes during the iterations to reflect the experience acquired by ants during the problem solving. It must be noticed that ants deposit an amount of pheromone proportional to the quality of their choice, the shorter the path, the higher the quantity of pheromone deposited This helps to direct search towards good solutions.

The main role of evaporation is to avoid stagnation. The memory of each ant is used to avoid doing a step which is not feasible. In the memory of each ant are stored the cities already visited, it is also called "tabu list". This memory also allows ants to cover the same path to deposit online delayed pheromone on the visited arcs.

The ant decision table $A_i = \left|a_{ij}(t)\right|_{|\aleph_i|}$ of node i is obtained by the combination of the local pheromone trail value and the local heuristic value:

$$a_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{i \in \aleph_i}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}} \forall j \in \aleph_i$$

where

- $\tau_{ij}(t)$ I s the amount of pheromone trail on the arc (i,j) at time t
- $\eta_{ij} = {}^{1}\!/_{d_{ij}}$ is the heuristic value of moving from node i
- N_i is the set of neighbours of node i
- α and β are parameters that control the relative weight of pheromone trail and heuristic value

The probability with which an ant k chooses to go from a node i to a node $j \in \aleph_i^k$ while building its tour at the t^th algorithm iteration is:

$$p_{ij}^{k}(t) = \frac{a_{ij}(t)}{\sum_{l \in \aleph_i} a_{il}(t)} \qquad (1)$$

where $\aleph_i^k \subseteq \aleph_i$ is the set of nodes in the neighbourhood of node i that ant k had not visited yet.

The parameters α and β have to be balanced correctly. If $\alpha = 0$, it is the closest cities which are more likely to be selected whereas if $\beta = 0$, only the pheromone amplification influences the choice what leads to a rapid situation of stagnation which usually gives a solution far from the best ones. Hence a trade-off between the influence of the two information, pheromone and heuristic must be done properly.

As said before, evaporation plays an important role in ACO algorithm. Before ants deposit their pheromone at the end of their tour, evaporation happens with an evaporation rate ρ . The amount of pheromone deposited on an arc (i, j) by an ant k at the end of the iteration by each ant is. $\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L^k(t)} & if \;\; (i, j) \in T^k(t) \\ 0 \; otherwise & \end{cases}$

Where $T^k(t)$ is the tour done by an ant k at the time t and $L^k(t)$ its length.

In practice the update of the pheromone trail for each arc is done accordingly o this t: $\tau_{ij}(t) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t)$ where $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$, m is the number of ants at each iteration.

Ant Colony System (ACS)

Dorigo and Gambardella proposed in 1996 an algorithm based on AS called Ant Colony System(ACS) [53, 66,63] as AS was able to find good solutions at the TSP only for small problems, they developed ACS in order to improve the performance of AS.

This algorithm differs on some points from the AS. The first difference concerns the update of the pheromone trail. The update done at the end of an iteration of the algorithm is called offline. Once all the ants have built a solution, pheromone trail is added to the arcs used by the ant that has found the best tour from the beginning of the trial. Thus instead of allowing all the m ants to update the pheromones, here only the ant that has found the best solution deposits pheromone on the arcs of the best tour.

The offline pheromone trail update rule is:

$$\tau_{ij}(t) = (1-\rho).\tau_{ij}(t) + \rho.\Delta\tau_{ij}(t)$$

Where $\Delta\tau_{ij}(t) = {}^{1}\!/\!_{L^+}$ and L^+ is the length of the best tour constructed since the beginning T^+ and $\rho \in [0,1]$ is a parameter governing pheromone decay. The update of the pheromone is applied only to the arcs (i,j) belonging to the best tour T^+.

The second difference concerns the decision rule in the construction of the solution. In ACS, ants use a so-called pseudo-random-proportional rule, in which an ant k on city i chooses the city $j \in \aleph_i^k$ to move as follows, if $A_i = \left|a_{ij}(t)\right|_{|\aleph_i|}$ is the ant decision table:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum\limits_{l\in\aleph_i}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}} \forall j \in \aleph_i$$

If we use q a random variable uniformly distributed over $[0,1]$ and if $q_0 \in [0,1]$ is a tunable parameter we can describe the random proportional rule used by an ant k located in node i to choose the next node $j \in \aleph_i^k$ is the following:

if $q \le q_0$ then $p_{ij}^k(t) = \begin{cases} 1 \; if \; j = \arg\max(a_{ij)} \\ \quad 0 \; otherwise \end{cases}$

else $q > q_0$ then $p_{ij}^k(t) = \dfrac{a_{ij}(t)}{\sum\limits_{l\in\aleph_i^k} a_{il}(t)}$

When $q \le q_0$ (probabilistic choice) the algorithm concentrates its activity on the best solution and when $q > q_0$ (deterministic choice) it concentrates its activity on the exploration of the search space. Thus it is possible to tune q_0 to moderate the degree of exploration and the degree of concentration on the best solution.

The third difference is that in ACS ants perform also a local update, called online step-by-step pheromone updates. They are performed to favour the emergence of other solutions than the best so far. An ant moving from the city i to the city $j \in \aleph_i^k$ updates the pheromone trail on the arc (i, j) according to the following rule: $\tau_{ij} = (1-\varphi).\tau_{ij} + \varphi.\tau_0$ where $\varphi \in [0,1]$.

When an ant moves from city i to city j, the application of the local update rule makes the corresponding pheromone trail τ_{ij} diminish. Thus arcs which are visited become less and less attractive. This facilitates the exploration of arcs not yet visited. Actually the more ants explore different paths, the more the chances are to find an improving solution if it is compared to the case where all the ants converge to the same tour.

The last difference concerns the utilization of a candidate list which provides additional local heuristic information. A candidate list contains a list of preferred cities to be visited from a given city. In ACS when an ant is in city i, instead of examining all the unvisited neighbours of i, it chooses the city to move to among those in the candidate list; only if no candidate list city has unvisited status then other cities are examined. The candidate list of a city contains cl cities ordered by increasing distance (cl is a parameter), and the list is scanned sequentially and according to the ant tabu list to avoid already visited cities.

Max Min Ant System (MMAS)

This algorithm is an improvement of the AS. It has been introduced in 1997 by Stützle and Hoos[146,147]. As in ACS, only the ant that has found the best solution is authorized to deposit pheromones on the arcs which constitute the best solution. It can be the best ant within iteration or the best ant since the beginning of the trial which contributes to the deposit. It is subject to the algorithm designer decision.

Another characteristic is that the pheromones trail values are restricted to an interval $[\tau_{\min}, \tau_{\max}]$ and that they are initialized to their maximum value $\tau_{\max}$. By putting explicit limits on the trail strength, it restricts the range of possible values for the probability of choosing a specific arc according to equation (1). One of the reasons why

Problem name		authors	year	References
Routing	Travelling salesman	Dorigo et al.	1991, 1996	[54]-[55]
		Dorigo & Gambardella	1997	[52]
		Bullnheimer, et al.	1997	[18]
		Stützle & Hoos	1997 , 2000	[146][147]
	Vehicle routing	Gambardella et al.	1999	[68]
		Reimann et al.	2004	[132]
	Sequential ordering	Gambardella &Dorigo	2000	[65]
assignment	Quadratic assignment	Maniezzo et al.	1994	[101]
		Gambardella et al.	1997	[67]
		Stützle & Dorigo	1999	[144]
	Course timetabling	Socha et al.	2003	[141]
	Graph colouring	Costa & Hertz	1997	[30]
Scheduling	Project scheduling	Merkle et al.	2002	[106]
	Total weighted tardiness	Den Besten et al.	2000	[36]
	Total weighted tardiness	.Merkle & Middendorf	2000	[104]
	Open shop	Blum	2005	[10]

Table 3: Non-exhaustive list of applications of ACO algorithm [49]

AS performed poorly when an elitist strategy, like allowing only the best ant to update pheromone trails, was used, is stagnation. Limited the values of the pheromone trail helps to avoid this phenomenon. But stagnation can also appear in MMAS in case some pheromone trails are close to $\tau_{\max}$ while most others are close to $\tau_{\min}$. Hence Stützle and Hoos have added what they call a "trail smoothing mechanism", pheromone trails are updated using a proportional mechanism: $\Delta\tau_{ij} \propto (\tau_{\max} - \tau_{ij}(t))$, the difference between the current quantity of pheromone and the quantity maximum. In this way the relative difference between the trail strengths gets smaller, which obviously favours the exploration of new solutions.

3.3.6 Applications of ACO algorithms

The TSP which is very often used to present a new ACO algorithm has already been described in the precedent section. But ACO algorithms are applied to many other combinatorial optimization problems with successful results. ACO algorithms have been applied to other NP-hard problems like Sequential Ordering Problem (SOP), Vehicle Routing Problem(VRP) and Quadratic Assignment Problem (QAP). In these cases, the ACO algorithm is very often coupled with a local search algorithm which takes the ants' solution to a local optimum before updating the pheromone trail. In Table 3 the main applications are listed with their references. And a more detailed description of some of then will be given afterwards.

Sequential Ordering Problem (SOP)

The SOP is a more general version of the asymmetric TSP, a TSP where the cost of going from a city i to a city j is different from the cost of going from a city j to a city i. The problem consists in finding a minimum weight Hamiltonian path[2] on a directed graph with weights on the arcs and on the nodes. The solution is subject to precedence constraints among nodes. SOP, which is NP-hard, models real world problems like single-vehicle routing problems with pick-up and delivery constraints, production planning, and transportation problems in flexible manufacturing systems . Thus the SOP is very important from an application point of view. In 2000 Gambardella and Dorigo proposed an extension of ACS, ACS-SOP which is associated with a local search. This method is called HAS-SOP (Hybrid Ant System for the Sequential Ordering Problem) [65], The only difference with ACS is that the solution given respects the precedence constraints HAS-SOP has been tested on different instances in all cases HAS-SOP was the best performing method in terms of solution quality and of computing time.

Vehicle Routing Problem (VRP)

In this problem, a set of vehicles parked in a depot has to serve a set of customers before returning to the depot. The aim is to minimize the number of vehicles and the total distance travelled by all the vehicles. A capacity constraint is imposed on the vehicle trips but other constraints such as time window, rear loading, maximum tour length or others deriving from the real world can be added. The basic VRP is the Capacitated VRP (CVRP).

ASrank, the rank-based version of AS, was applied to this problem by Bullnheimer, Hartl and Strauss applied an AS rank algorithm, a rank based version of AS to this problem and obtain very good results. [41,42]**.**

According to the good results obtained by ACO algorithm for SOP and CVRP, ACS was applied to a VRP with time window. In this problem, called VRPTW, a time window $[b_i, e_i]$ is linked to each customer i. The customer must be served in this interval of time. The objective is first to minimize the number of tours (or vehicles) and then minimize the total travel time. A solution which lowers the number of tours or vehicles is always preferred to a solution with higher number of tours or vehicles even if the total travel time is higher. This problem is very often studied in the literature and both objectives can be

[2] a Hamiltonian path is a path in an undirected graph which visits each vertex exactly once

antagonistic in case constraints are very tight (for example when the total capacity of the minimum number of vehicles is very close to the total volume to deliver to the customers or when time windows are narrow, minimizing the total travel time can include a higher number of vehicles [94].

Gambardella et al. in 1999 [71] have designed an ACO algorithm based on ACS for Multi objective VRPTW. This Multiple Ant Colony System (MACS) is organized with hierarchical colonies. One colony (ACS-VEI) is designed to minimize the number of vehicles while the other one (ACS-TIME) has for objective the minimization of the total travel time. Both colonies use independent pheromone trails and they collaborates by exchanging information during the update of the pheromone. Both colonies use independent pheromone trails but they collaborate by exchanging information through mutual pheromone updating. In the MACS-VRPTW algorithm both objective functions are optimized simultaneously: ACS-VEI tries to diminish the number of vehicles searching for a feasible solution with always one vehicle less than the previous feasible solution. ACS-VEI is a little bit different from the traditional ACS, when ACS has for best solution the shortest tour, ACS-VEI has the tour (usually unfeasible) which has the highest number of visited customers. In the contrary, ACS-TIME is a traditional one, it is used to minimize the travel time of the solution found by ACS-VEI. For the update of the pheromone of the ACS-VEI, they use a combination of the best solution found by the two algorithms for better results. This algorithm has been experienced as the most effective.

The algorithm developed by Gambardella has been used for industrial application by the intermediary of Antoptima. Antoptima is a spin-off of Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), a leading Research Institute in Artificial Intelligence. It displays tools for the solution of VRP with algorithms based on ACO. The first tool proposed, is called DYVOIL, is a software application for the management and optimization of heating oil distribution. The second tool AntRoute, is a software for large scale dynamic optimization of vehicle routes and fleets. This software is used by Migros, the main Swiss supermarket chain and Barilla, the Italian pasta maker.

Quadratic assignment problem (QAP)

The quadratic assignment problem (QAP) is the problem consists in assigning n facilities to n locations with the objective of minimizing the cost of the assignment, where the cost is defined by a quadratic function. This problem can be solved with optimality only for small instances and it is considered as one of the hardest combinatorial problem. Several ACO algorithms have been proposed to attack the QAP, from the basic AS to more advanced version. [100,101].

Two efficient algorithms which have been applied to this problem are MMAS -QAP [144] and [HAS-QAP [67]. Both algorithms have been tested and it has appeared that the performance strongly depends on the type of instances. Taillard has categorized the instances:

1. Unstructured uniform random
2. Unstructured grid distance
3. Real world
4. Real world like

AS-QAP performs well for real world irregular and structured problems but it is less competitive for unstructured, random and regular problems. ANTS another ACO algorithm does not suffer from this dependency to the type of problem.

This problem-dependency was not shown by ANTS, which was also applied to QAP. In order to apply ANTS to QAP, it is necessary to specify the lower bound to use and what is a move in the problem context. The application described in [100] such definitions are done. As for the lower bound, since there is currently no lower bound for QAP, which is both tight and efficient to compute, the LBD bound was used. As for the moves, it was declared that a move corresponds to the assignment of a facility to a location, thus adding a new component to the partial solution corresponding to the state from which the move is originated. Some considerations on the move structure were used to improve the computational effectiveness of the resulting algorithm. ANTS was tested on instances up to $n = 40$ and showed to be effective on all instance types; moreover its direct transposition into an exact branch and bound was also effective when compared to other exact algorithms.

Scheduling problem

The general approach to solve scheduling problems such as the Single Machine Total Weighted Tardiness Problem (SMTWTP) or the Permutation Flowshop Scheduling Problem (PFSP) is well defined. Starting with the first place of the schedule, at each iteration, every ant decides which job has to be put at the next place in the sequence. As said before the pheromone trail τ_{ij} refers to the desirability of putting the job i in the position j if the objective is to minimize the makespan or the total tardiness and its weighted variants. This approach is natural since for many PFSP it exists a good list of scheduling heuristics information which can be used by the ants in addition to the pheromone information.

One of the first works using ACO was carried out by Dorigo et al. in 1994 [28]. In this study they deal with a job shop scheduling problem. In 1999 van der Zwann and Marquez also produced an ACO algorithm for a jobshop problem [153]. Another scheduling problem regularly studied in the literature is the single machine total tardiness problem (SMTTP) and its weighted variants (variant [3,36,37,104]. A comparison between ACO and other methods has been undertaken and has shown that ACO algorithms give better results than best known methods for SMTWTP. Gagne et al. have also used an ACO approach to deal with scheduling problems taken from the industry [61].

If we consider PFSP, Stützle has applied its MMAS algorithm in combination with a local search to a PFSP with the objective of minimizing the makespan. [142]. This procedure was tested on the 90 Taillard benchmark permutation flowshop p gives high quality solutions in short time and a comparison to state-of-the-art algorithm shows that it performs better or at least gives comparable results. This first approach to FSP using ACO algorithm gave very promising results.

In 2001, Rajendran and Ziegler [129] have also proposed their ACO algorithm First they proposed an improvement of MMAS by integrating the summation rules suggested by Middendorf [105] and using their own local search. This algorithm is called M_MMAS. The summation rule suggests a modification with respect to the selection of the job to be appended to the partial ant-sequence. When choosing a job i to put at a position j, instead of considering only the quantity of the pheromone with respect to the position j, the choice is based on the pheromone value up to the position j. This summation value of pheromones is an indicator of the need and the desire of placing a job i not later than the position j. Another difference is that the choice of the job is made among the first five and not all unscheduled jobs.

They also proposed another ACO algorithm called PACO. This algorithm is applied after having used the NEH algorithm in combination with a local search to construct a solution which is indeed of good quality. This solution is used for the initialization. The differences with classical ACO concern first the initialisation of the pheromone trails, secondly the construction of the solution and thirdly the way pheromone trails are updated.

In PACO, the initialisation of the pheromone matrix is done accordingly to the position of the job in the sequence given by NEH heuristic. This is done so that the influence of a good seed sequence is more reflected in the pheromone matrix, the value of the pheromone is not the same for all the elements of the matrix. τ_{ij}, the desirability of setting a job i at the position j is higher if the position of job i in the sequence given by NEH is closed to the position j.

While ACS offers 2 possibilities as to which job to place in which position, PACO ads a third pick to the choice. In the construction phase, instead of having two different types of choice as in ACS, a third possibility exists, choosing the first unscheduled job in the best sequence obtained so far.

Finally, the update of the pheromone trails. The update is based on the relative distance between a given position and the position o the job in the resultant sequence. The idea is to deposit more pheromone for the jobs occupying a position which is closer to its position in the best sequence obtained so far. The authors say that performance is better than the ant approach described by Stützle in 1998.

As for the VRP, Industrial applications have already been developed in the scheduling domain. Eurobios (www.eurobios.com) uses an ACO algorithm for a continuous two stage flowshop problem with finite reservoirs for example. Real world constraints such as setup times, capacity restrictions, etc. are taken into account in the application of the algorithm.

Dynamic problem

In their natural environment, ants are able to react quickly at any change in their environment. If an obstacle appears on their current shortest path to the food, they quickly find a new path which will become after a while the new shortest one. Hence dynamic combinatorial problem is a logical application for ACO algorithms. In dynamic problems, the search space changes with time. The conditions of the search, the definition of the problem is not stable. Thus the quality of a solution found varies with time. Thus to solve this kind of problem, the algorithm must be able to adjust the search direction to each new environment.

In the literature lots of work uses ACO algorithm to tackle communication routing problem [48,52]. In networks problem, the cost of components or of connections can change over time.

The generic routing problem in communications networks consists in building and using routing tables to direct data traffic so that an objective (bandwidth, delay), measure of performance is optimized. For this kind of problem, ants are launched from each node of the network and travel through the network by applying a probabilistic transition rule based on pheromone and sometimes heuristic or local information.

ACO algorithms are divided in 2 classes:

- the connection-less networks: data packets of a same session can follow different paths
- the connection-oriented networks: all the data packets of a same session follow the same path selected in a preliminary setup phase

Schoonderwoerd, Holland, Bruten and Rothkrantz have made in 1996 the first attempt to apply ACO algorithm to routing problems, this algorithm is called ABC [138,139]**.** Two years later, Di Caro and Dorigo have proposed an ACO algorithm called AntNet [42,43] which outperformed a number of state-of-the-art routing algorithms for packet-switching networks on a set of benchmark problems.

Other problems like dynamic version of the TSP. In the dynamic version, the distance between two cities changes or some cities are added and removed. [57] Real dynamic VRP has also been tackled by an ACS algorithm with good results [68].

3.3.7 Conclusion

ACO is a metaheuristic like genetic algorithm, simulated annealing or tabu search which is inspired by particular natural phenomenon. Based on simple principles, real ants' behaviour can be enriched with new artificial capacities and specific problem information so that they show very good performance and sometimes world class performance for many applications such as QAP with AS-QAP, HASQAP or MMAS-QAP or network routing with AntNet.

An ACO algorithm is one of the most successful applications of swarm intelligence, a field characterized by stigmergetic model of communication, an indirect communication which has all its importance in the success of ACO. This success for lots of academic problems and real industrial applications explains why hundred of researchers worldwide work on applying ACO to classic NP-hard optimization problems. Other works concerns the application of ACO to dynamic and multiple objectives problem. This work deals with this last field, multiobjective optimization.

4

Multiobjective optimization

In many sectors of the industry (mechanical, chemistry, telecommunication, environment, transport, etc.), optimization problems that arise are complex and of great dimension. They have to be optimized, but they are never or rarely single. Usually several objectives often conflicting have to be taken into account. For example, maximizing the quality and minimizing cost of a product; maximizing the speed and minimizing fuel consumption of a vehicle; minimizing weight and maximizing the strength of a particular component, etc.

All the objectives have to be satisfied simultaneously and a good trade-off solution has to be found. An optimized solution according to one objective often implies poor results for one or more of the other objectives. Thus the problem is to find a compromise, a solution which has acceptable performance for all the different objectives. Acceptable performance is most of time sub-optimal in the single objective sense. Thus instead of finding one optimal solution, multiobjective optimization is characterized by a family of solutions which are considered as equivalent on the absence of any information concerning the relevance or the importance of one objective relative to the others.

Multiobjective optimization is a discipline which deals with this kind of problem. The origin of this discipline comes from a work in the economy by Edgeworth and Pareto [123]. Different techniques exist to solve this kind of problem. On one hand exact methods, only feasible for problems of small size; on the other hand approximate methods for more complex problems. Before presenting a brief review of the different methods used to solve Multiobjective problems (MOP), we will first give a more formal definition of the MOP and of some associated concepts.

4.1 Definition

In a formal way, a Multiobjective optimization problem (MOP) can be defined as follow:

$$(MOP) = \begin{cases} Min(F(x)) = (f_1(x), f_2(x), ..., f_n(x)) \\ x \in C \end{cases}$$

where:

$n \geq 2$ is the number of objectives,

$x = (x_1, ..., x_k)$ is the vector representing the decision variable. In the case of Multiobjective combinatorial problem (MCOP), the vector $(x_1, ..., x_k)$ has a finite number of possible values.

C represents the set of solutions realizable according to all the existing constraints

$F(x)) = (f_1(x), f_2(x), ..., f_n(x))$ is the criteria vector to be optimized.

In the ideal case, it exists a vector $y^* = (y_1^*, y_2^*, ..., y_n^*)$ which optimizes each objective function f_i, $y_i^* = \min(f_i(x)), x \in C$. But, unfortunately, objectives are often conflicting in real problems and such vector y^* cannot be found.

Thus another concept has been introduced for solving multiobjective problems. This is the concept of Pareto dominance.

For a minimization problem, a solution $y = (y_1, y_2, ..., y_n)$ dominates a solution $z = (z_1, z_2, ..., z_n)$ if and only if $\forall i \in [1...n] \mid y_i \leq z_i$ and $\exists\ i \in [1...n] \mid y_i < z_i$. It is clear that if a solution *A* dominates a solution *B*, *A* is better than *B*.

The second concept introduced is that of the Pareto optimality, a solution $x^* \in C$ is Pareto optimal if and only if there does not exist a solution $x \in C$ such as $F(x)$ dominates $F(x^*)$. Pareto optimal solutions are also called non dominated or efficient solutions.

The solution of a MCOP is the set of Pareto optimal solutions. These solutions form what can be called the Pareto front. Any solution of this set is optimal in the sense that no improvement of one component of the objective vector can be made without

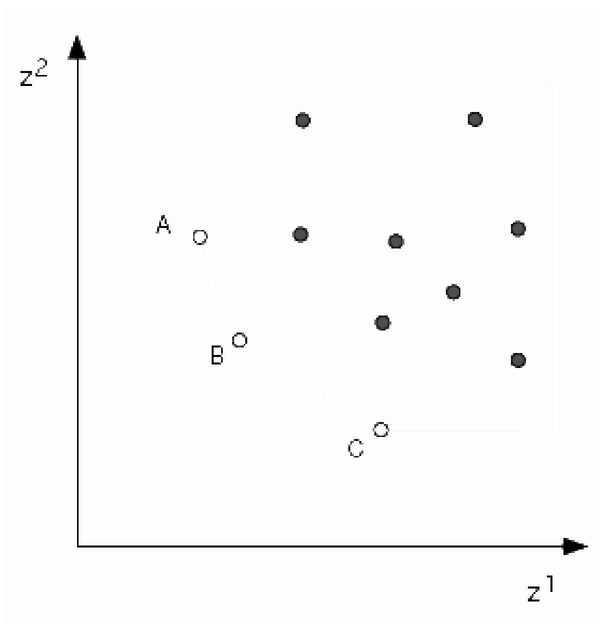


Figure 19: Illustration of Pareto dominance [69]

degradation of at least one of the other components of the vector function. In Figure 19, the points A, B, C constitute an approximation of the Pareto front, they are not dominated by any other points.

Actually for real problems, determining the Pareto front or an approximation of the Pareto front constitutes the first step of the solving process of a multiobjective problem. In practical MCOP, the determination of the Pareto optimal set is only the first phase. In a second phase, the decision-maker has to choose in this set of solutions the solution that satisfies him according to its preferences, the problem environment and the knowledge he has of the problem. Hence multiobjective optimization has for purpose to facilitate the task of the decision maker by limiting the number of possible solutions to the "best" ones.

The role of the decision maker in the formulation of a multiobjective optimization problem is crucial. He can have preferences for one or several objectives and he has often a good knowledge of the problem. Depending on his knowledge, different approaches to solve the problem can be taken.

The problem can be treated as a single objective problem if the decision maker has a very good knowledge of the problem whereas a Pareto approach will be chosen if he prefers to make its choice among a set of different solutions that are opposed to him.

4.2 Techniques of optimization for Multiobjective combinatorial problem

Large numbers of research efforts have been dedicated to multiobjective optimization. There are three main categories in which can be classified the different techniques of resolution of MCOP:

– scalar approaches: the MCOP is transformed into a single objective problem, they are algorithms based on aggregation which combines the different objective functions f_i into one function F. To apply this kind of approach, the decision maker must have a very good knowledge of the problem.

– Pareto approaches: the solutions are generated according to the concept of non dominance.

– non-Pareto and non-scalar approaches: there is no transformation into a single objective function, they use operators to attack the different objective functions separately.

The three approaches will be now briefly described.

4.2.1 Scalar approaches

Different methods can be used to transform the MCOP into a single objective problem. This kind of approach is very popular for its low computation cost and its simplicity but it gives only one Pareto optimal solution. In order to find more Pareto optimal solutions, the algorithm must be run many times with different values of the parameters, what can lead to a dramatical increase of the computation time.

Aggregation method:

This is the first method and the simplest method used to tackle MCOP by aggregating all the functions f_i into one function generally in a linear way by using a weight vector $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$, where λ_i is the weight associated to the objective function f_i, $F = \sum_{i=1}^{n} \lambda_i . f_i$ [83]. This simple approach has been used in different metaheuristics such as genetic algorithm [149] or simulated annealing [140] and tabu search. [33].

In order to obtain a set of non dominated points, an aggregating method can be applied multiple times with different values of the weights .Then, the search directions are dynamically modified during the search process. It is suggested in [86] that these methods perform very well

ϵ–constraint method:

Here an objective function f_i is optimized but with constraints linked to the other objectives functions. This approach has been made with genetic algorithms [127], tabu search [79] or hybrid metaheuristics [128]. It is possible to generate different Pareto optimal solutions by changing the values of the constraint ϵ.

Goal programming:

The decision maker defines the goals to reach for each objective. These values are introduced into the formulation of the problem to transform it into a single objective problem. For example the objective could be, with the integration of a weighted norm in the cost function, the minimization of the deviation from the goals. Different works using genetic algorithm [25,89], simulated annealing [140] or tabu search [70] exist.

4.2.2 Non-Pareto/non-scalar approaches

This approach is based on a population solutions and the search is carried out by treating the different objectives separately.

Parallel selection:

Schaffer [137] has developed a genetic algorithm called VEGA (Vector Evaluated Genetic Algorithm). This technique uses what can be called a parallel selection during the selection phase. Individuals are selected from the population according to each objective independently from the others. Thus he works with a number of subpopulations equal to the number of objectives. During the reproduction phase, the algorithm composes the entire population by using the traditional operators (crossover and mutation).

Lexicographic selection:

In this approach the different objectives are classified in an order of importance by the decision maker. Then, the search is carried out according to this order which defines the significance level of the objective functions. Gravel et al. have developed a method to solve multiobjective problems based on this approach [62].

4.2.3 Pareto approaches

By contrast with the two first approaches where the objectives are treated separately or where a utility function is used, this approach uses the concept of Pareto dominance as an acceptance criterion. Goldberg [73] was one of the first to use this concept with a genetic algorithm.

The advantage of this pure Pareto approach is that it can generate Pareto optimal solutions in the concave portions of the front. Scalar approaches generate only supported solutions if the scalarization is optimally solved.

Zitzler and Künzli [156] have introduced IBEA (Indicator Based Evolutionary Algorithm), a method presenting a new idea of Pareto for Evolutionary algorithms. The idea is to use a binary performance measure which can be based on the decision maker preferences, to compare a pair of solutions. Then the selection can be made according to this performance measure. In these methods, a rank is usually assigned to the different non dominated points [59,153]

4.3 Metaheuristics for multiobjective optimization

Many algorithms and methods which have been successfully applied to single objective problems have been extended to multiobjective optimization problems. In this section, some of these methods like local search, tabu search, genetic algorithms or ant colony optimisation will be briefly presented.

4.3.1 Tabu search

MOTS (Multiple objectives Tabu search) was proposed in 1997 by Hansen to generate non dominated solutions to MCOP [78]. In the procedure, the set of current solutions are optimized through manipulations of weights towards the Pareto front while at the same time the algorithm tries to disperse them over the Pareto frontier.

Another procedure uses a set of non dominated solutions of good quality and diversity found by an evolutionary algorithm and then applies to each solution a local search for which an objective function needs to be defined. The objective functions defined are such that two search made simultaneously do not explore the same area of the search space. Hence the search is intensified around the solutions found and this without lost of diversity. This algorithm called Target Aiming Pareto Search (TaPaS) is described in [88].

Other tabu search algorithms have been proposed in the literature. Most of time, they do not use a total Pareto approach to generate the initial solution. Hertz [79] proposed three different non Pareto approaches (weighted, ϵ-constraints and lexicographic). Beausoleil proposed in [7] a weighted objective Tabu search to generate the initial solution.

4.3.2 Genetic programming

An extended genetic algorithm called Multiple Objective Genetic Programming (MOGP) was proposed in 1997 by Rodriguez-Vasquez. Genetic programming has a representation of the chromosomes in a hierarchical tree, what can be more powerful in some situations. Bleuer et al. have applied a genetic programming algorithm to a problem where the goal is to evolve compact programs and to reduce the effects caused by bloating. Two independent objectives are considered, the program size and the program functionality. By associating genetic programming and SPEA2, a genetic algorithm [157], they obtain an algorithm with good performance. In their studies, one objective is solved as a single objective problem by a genetic programming method while the second is solved with a multiobjective evolutionary algorithm.

4.3.3 Simulated annealing

Lots of works using SA for MOP can be found in the literature Most of these algorithm store non dominated solutions found during the search process in an archive [25]. They rarely use the concept of Pareto ranking, more often the acceptance function is an aggregation of the different objectives functions made with a weight vector.

4.3.4 Ant colony optimisation

Several Multiobjective ACO approaches (MOACO) can be found in the literature. They can be Pareto or non Pareto approach according to the solution they give at the end of the process. In the following, some non Pareto approaches will be presented and then other, Pareto approaches like MOAQ, Bicriterion ant, P-ACO and COMPETants, will be briefly presented. Most of them work with multiple colonies and/or multiple pheromone matrices and usually for the Pareto approach, non-dominated solutions found during the process are saved in an archive. For non Pareto approaches, the decision maker has usually given an order of preference for the different objectives, what influences the search procedure of the algorithm.

Here are some non Pareto approach algorithms that have already been developed.

Multiple Ant Colony System for Vehicle Routing Problem with Time Windows

A biobjective vehicle routing problem has already been described in the section 3.3.6.

Multiple Objective Ant Colony Optimizations Metaheuristic

Gagné et al. in [61] have tested a bicriteria approach of a single machine total tardiness problem with changeover costs and two other criteria. In this problem, the objective changeover cost is more important. In fact ants take into account all the objectives during their decision process, but the quantity of pheromone deposit depends on the value of the changeover cost of the solution.

Gravel et al. have also proposed a method to solve multiobjective real-world scheduling problems related to aluminium production industry [62]. In this method, the objectives are lexicographically ordered by the decision maker. The algorithm, named MOACOM, has not the aim to provide a set of good non dominated solutions. Only the best solution according to the lexicographic order is taken into account.

SACO

This non Pareto approach has been specifically designed by T'kindt et al [150] to solve a 2-machine bicriteria flowshop scheduling problem. The characteristic of this algorithm is a stronger diversification at the beginning of the search and intensification in the following. This method deals with one single solution, the solution having the best cost for one of the objectives. As in ACS, the ant chooses between intensification and diversification according to a tunable parameter.

We will now present some ACO algorithms using a Pareto approach.

Multiple Ant Colony System

Barán and Schaerer introduced Multiple Ant Colony System (MACS) [6]. This variation of the algorithm proposed by Gambardella for the VRPTW uses a single pheromone matrix .When a solution has been generated, it is compared to the solutions of the set of non dominated solutions found so far and then the Pheromone trails are updated accordingly to a function of the value of the non dominated solutions for each objective.

Multiple Objective Ant-Q Algorithm

In 1999, Mariano and Morales have proposed an ACO algorithm based on the algorithm ANT-Q [64] named Multiobjective ANT-Q (MOAQ) [102], where they use different ant colonies. Each colony is associated to an objective. This technique has been applied to the design of a water distribution irrigation network problem with multiple objectives. In this algorithm each colony is under the influence of one part of the solution found by the previous colonies accordingly to the other objectives. One ant from colony i receives a part of the solution of the colony $i-1$ and tries to improve it according to criterion i. In this algorithm, a value r called reward, which models how good an action helps to find trade-off solutions, is used to reinforce path what lead to better solutions.

Each solution that has visited all the colonies is compared the ones which belong to the set of non dominated solutions found in the previous iterations. All the non dominated solutions found along the process are stored in an archive.

Ant Algorithm for Bi-criterion Optimization Problems

In [84], Iredi et al. have developed an algorithm called BicriterionAnt to solve a biobjective scheduling problem. They associate one type of pheromone to each objective. When all ants have generated their solution, all non dominated solutions are allowed to update the pheromone trails proportionally to the number of ants which are non dominated and to the quality of their solution.

In the same work, they proposed another algorithm called BicriterionMC which differs from BicriterionAnt in the update one of the pheromone matrix .They consider two methods of pheromone update:

- Update by origin: here each ant updates the pheromone matrix of its own colony, what enforces both colonies to search in different regions of the Pareto front
- Update by regions: the Pareto front is split into 2 parts of equal size, the ant which has found a solution in the ith part updates the colony i. This helps to guide the colonies to search in different regions of the Pareto front, each of them in one region.

As in MOAQ, non dominated solutions are saved in an archive.

Pareto Ant Colony Optimization

In order to solve a multiobjective portfolio selection problem, Doerner et al. have proposed an algorithm called Pareto-ACO (P-ACO) [44,45]. It uses ACS but with a difference in the pheromone update. Only the best and the second best solutions generated in the iteration for each objective k is used for the update. They use one pheromone matrix for each objective and each ant, at each iteration generates a weight vector which is used to aggregate the different objective;. Once again on dominated solutions are saved in an archive.

COMPETants

Doerner et al. in [46,47] propose to solve a biobjective transportation problem. In [46], they used an algorithm based on the ACO algorithm AS rank-based [18]. It is called COMPETants and it uses two colonies, one for each objective. They are used to solve the problem. Each colony uses its own pheromone matrix. When every ant has built its solution, solutions are compared and the best colony receives more individuals for the next iterations. Another characteristic is that both colonies collaborate thanks to the use of special ants called "spy" which combine the pheromone information of the two colonies.

For the same kind of problem but with objectives of different importance, they have used another approach. In [47], one colony, the "master" colony is associated to the most important objective and the "slave" colony to the other. All the k iterations, the "master" colony updates its pheromone matrix accordingly to the solutions found by the "slave" colony. This approach is a lexicographic approach similar to what is done in MACS where one objective is more important than the other.

PACO

Guntsch and Middendorf have proposed an algorithm called PACO (Population based ACO) in [75,76]. This algorithm differs from the standard ACO algorithm in the way pheromone trails are updated. Usually pheromone trails have first a negative update, the evaporation and then a positive update with the best solutions found by the ants. In PACO for single objective approach, a set of k best solutions is used for the update. If a solution enters this set, there is a positive update of the pheromone trail whereas if a solution is removed, there is a negative update. In the multiobjective approach, they introduce "the Average-Weight-Rank", a method for constructing the selection probability distribution for the ants and the new derivation of the active population to determine the pheromone matrices. They have applied this algorithm to a single machine total tardiness with changeover costs problem.

4.3.5 Hybrid metaheuristics

In the literature, articles often combine genetic algorithms with local search [86] this kind of algorithm is also called memetic. The principle consists in incorporating local search during the GA search. There are several ways to incorporate it in the genetic algorithm search. Local search can replace the mutation operator, or it can be applied after having created each new generation. All have in common that a local search is used to improve individual solutions

Ishibuchi [85] has developed a memetic algorithm which was used to attack a biobjective flowshop problem. Another memetic algorithm called Adaptive Genetic/Memetic Algorithm (AGMA) is described in [15]. In this method, memetic and genetic algorithms are hybridized. Another hybrid approach combining memetic algorithm, local search and path relinking has been developed to solve a biobjective flowshop problem [14]. In [89], we have a hybrid algorithm combining a genetic algorithm and a tabu search using the Target Aiming Pareto Search Principle (TAPaS). The search goals are defined according to the shape of the current set of Pareto solutions

A hybrid algorithm based on simulated annealing which has been applied to a biobjective space allocation problem has been described in [20]. Another new approach is to associate multiobjective metaheuristics with exact methods. In [150], a biobjective flowshop problem is solved. In this problem objectives are ordered lexicographically. One objective is not NP-hard and thus can be solved exactly while the other is solved by an ant colony algorithm.

In [88], a biobjective routing problem is solved. The algorithm used a genetic algorithm in cooperation with a branch & cut algorithm, this last algorithm is used to solve exactly one of the two objectives considered.

4.3.6 Parallel algorithms

Implementation in parallel of algorithms has also been used to tackle multiobjective problems. These approaches are rarely considering the concept of Pareto optimality in their design. One example is the island model where one objective is treated per island [98,133] or where each island has different aggregation weights. Parallel implementations have been classified in [31,32]. Two main different strategies exist. On one hand, there is the single walk parallelization where the objective is to speed up the sequential algorithm, on the other hand, there is the “multiple walk parallelization where the objectives are both a speed up and an improvement of the solution quality.

Single walk parallelization

The goal here is only to speed up the computations and to leave the basic behaviour of the algorithm unchanged. It is the simplest and the most used parallelization method for MCOP. MCOP. Real applications are usually problems of large size and large amount of computation time, thus, a parallelization of the search operators or of the evaluations of the objectives functions can help to speed up the algorithm.

Multiple walk parallelization

This type of parallelization has for first goal an improvement of the solution quality and then to speed up the algorithm. In [97], it is possible to find one of the first attempts to apply this kind of algorithm to multiobjective optimization. In this approach, it exists two different ways to build the Pareto front:

– centralized Pareto front (CPF): at the end of the process, the set of Pareto optimal solutions is constituted of global Pareto optima. Actually the Pareto front is built by the search threads during all the computation [15,27].

– distributed pareto front (DPF): the set of solutions is constituted of locally optimal solutions. Thus after having worked with locally optimal solutions, it must combine these solutions at the end of the process [134,5,111].

Actually it is important to notice that CPF implementations are constituted by different DPF which provides local optima front. Then these solutions are combined to form a single optima Pareto front.

Most of works deal with genetic algorithm, but some of them deal with tabu search [4] or ant colony optimization [38].

4.4 Two different ACO approaches for a biobjective flowshop problem

In this work we propose two ACO algorithms (*1phero* and *2phero*) to tackle the biobjective permutation flowshop problem. Different variants and configurations of these algorithms will be tested and compared. We have already reviewed that multiobjective approach with ACO uses one or multiple colonies and with one or multiple pheromone matrices.

The two proposed approaches use multiple colonies and the idea is to force each ant colony to search in different regions of the non dominated front. One method consists in aggregating the objective functions into one single objective and then dynamically modify the search direction during the search process. The second approach aggregates two pheromone matrices, each of them associated to one objective and the solution will be constructed with this aggregated matrix. The motivation for these approaches is to exploit the effectiveness ACO algorithms for single objective problems.

The underlying idea of the first approach is to solve a biobjective problem by aggregations of the two objectives into a single-objective one. It is less clear why it is recommended to aggregate pheromone matrices to tackle the biobjective problem. We know that performance of ACO algorithms for single objective problems tackling each objective separately, are very good. But we do not know what will be the behavior of the algorithm with such transformations. Anyway, as this method has shown good results in other works, it can be useful to compare the two approaches for a biobjective flowshop problem.

The aggregation into a single objective is based on a normalized weight vector, we have $\lambda_1 + \lambda_2 = 1$ and $F = \lambda_1.f_1 + = \lambda_2.f_2$ where F is the function used to rate the solution, f_1 and f_2 are the two objective functions to optimize. The same kind of aggregation is used for the pheromone matrices, $\tau_{ij} = \lambda.\tau_{ij}^1 + (1-\lambda).\tau_{ij}^2$ where τ_{ij} is the matrix used in the construction of the solution and τ_{ij}^1 and τ_{ij}^2 the pheromone matrices associated to f_1 and f_2. In the extreme cases, with $\lambda_1 = 1$ and $\lambda_1 = 0$, the process consider only the first objective and with $\lambda_1 = 0$ and $\lambda_2 = 1$, only the second objective is considered.

```
-¨ $F=\{f_1,f_2\}$: objective functions
-T: pheromone matrices
$W=\{\lambda_1,\lambda_2,...,\lambda_n\}$: weight vector
$s_0^*$: initial sequence
$A$: archive of non dominated solutions
Procedure 1phero
    for( $i=1,...,n$ )
        $F_i$ =Modify F( $\lambda_i$, $F$ )
        $N^*$ =GenerateSolutions ( $F_i$, $T$, $s_{i-1}^*$ )
            Construction($T$)
            Local search( $F_i$ )
            Update ($T$, $F$, $A$)
        end GenerateSolutions
        $A$ =UpdateArchive( $N^*$ )
    (optional( $s_i^*$ =SelectInitialSolution( $A$ or $N^*$ ))
    end for
end 1phero
```

Figure 20: Procedure of 1phero algorithm

```
-¨ $F=\{f_1,f_2\}$: objective functions
-$T=\{\tau_{ij}^1,\tau_{ij}^2\}$: pheromone matrices
$W=\{\lambda_1,\lambda_2,...,\lambda_n\}$: weight vector
$s_0^*$: initial sequence
$A$: archive of non dominated solutions
Procedure 2phero
    for( $i=1,...,n$ )
        $F_i$ =Modify F( $\lambda_i$, $F$ )
        $\mathrm{T}_i$ =ModifyTau( $\lambda_i$, $\mathrm{T}$ )
        $N^*$ =GenerateSolutions ( $F_i$, $\mathrm{T}_i$, $s_{i-1}^*$ )
            Construction( $\mathrm{T}_i$ )
            Local search( $F_i$ )
            Update ( $\mathrm{T}_i$, $F$, $A$)
        end GenerateSolutions
        $A$ =UpdateArchive( $N^*$ )
    (optional( $s_i^*$ =SelectInitialSolution( $A$ or $N^*$ ))
    end for
end 2phero
```

Figure 21: Procedure of *2phero* algorithm

Thus we use heterogeneous colonies where each ant of a colony weights the two objectives differently with different values of λ. During all the process, for all the different values of λ , all the non dominated solutions found will be saved in an archive and this archive will constitute the approximation set at the end of the procedure. The procedures of two algorithms are summarized in Figures 20 and 21. The details of the different functions will be given in the following.

In the following, we will present a local search introduced for the multiobjective optimization problem. The details of the different algorithms, the different variants and configurations tested will also be given.

4.4.1 Non dominated local search

We also use a local search based on the notion of dominance and on insert for the definition of the neighbourhood,. The non dominated local search looks for non-dominated solutions in the insert neighbourhood of the current point by moving each job of the sequence to each possible position and saves it in an archive if it finds one.

This function is applied once after having constructed a solution and having improved its quality with insert. It allows to generate more non dominated points and may be points which belong to the concave part of the Pareto front. We will refer to this function by the notation ND_LS.

4.4.2 ACO algorithm for biobjective problems

We have already presented the general procedure of our two biobjective ACO approaches (see Figure 20 and 21). We will now present each function separately. We will first present different strategies for the aggregation of the objectives and the matrices, and then the two different ACO approaches and their variants.

$W = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$: Aggregation of *F*

The modification of the aggregated objective function *ModifyF* is carried out by changing the weights assigned to each objective. Different strategies are possible; the change can be gradual or random, depending on how good solutions are placed in the search space. If solutions are clustered in the search space, a gradual change

$\lambda_i = \lambda_{i-1} \pm \frac{1}{(n-1)}$ should be preferable but if good solutions are spread all over the search space, a random change may be more useful.

The number of aggregations also plays a role in the performance of the algorithm. High number of weight combinations should return a better approximation of the Pareto front. But a high number of aggregations also leads to a large increase of the computation time. Thus a trade-off must be found between the number of aggregations and computation time.

We have tested different direction changes and different number of aggregations for different approaches, the results will be presented in a following section.

4.4.3 ACO algorithm using one pheromone matrix (1phero)

In this section, we present in details the different functions which constitute the *1phero* procedure and the two different variants on the initialisation of the solution.

The first possibility for the initialisation of the solution is to start from scratch for each value λ_i or to use a *2phase* approach.

1phero scratch

With this approach all the different colonies work without collaboration. Then new searches for the different weights will not be influenced by the solutions found for the previous weights, what allows a larger exploration of the search space.

1phero 2phase

Here, the solution found for the previous weight, λ_{i-1} is used for the initialisation of the current value of aggregation λ_i. Thus at each aggregation weight, the procedure starts with a solution which may be close to a good solution for the aggregation value. This is possible because the change in the weight vector is only minor. What is essential in *2phase* approach is that aggregation and g*enerateSolutions* are treated like a chain. Depending on the variant chosen, the last function of the procedure, *SelectInitialSolution* is applied or not. If it is not applied, we use the initial sequence s_0^* for the initialisation for each different weight.

F_i =Modify F(λ_i , F)

The two objective functions are simply aggregated with the factor λ_i .

$F = \lambda.f_1 + (1-\lambda_i).f_2$ where f_1 and f_2 are the two objective functions to optimize

GenerateSolutions (F_i , T_i , s^*_{i-1})

This function is divided in three operations:

- construction of a sequence with an ACO algorithm
- local search :
    - insert
    - ND_LS
- update of the pheromone trail

UpdateArchive(N^*)

The purpose of this function is to add to the archive A all the solutions $s \in N^*$ which are not dominated by any points $a \in A$. The second phase of this function is to filter A , this function deletes all dominated solutions in A and returns the filtered set.

s_i^* =SelectInitialSolution(N^*)

This function is applied when the variant *1phero 2 phase* is chosen. The same solution s_i^* which is allowed to update the pheromone is chosen for the initialisation of the next aggregation *GenerateSolutions* function. In *1phero scratch*, this function is not used.

5.4.4 ACO algorithm using two pheromone matrices (2phero)

As in *1phero*, two variants are possible, *scratch* and *2phase* approach. One of the difference in *2phero* is that two strategies are possible for the update of the pheromone matrices and for the selection of the initialisation solution. Here will make the difference between a *global* strategy (*2pheroG*) and a *local* strategy (*2pheroL*).

In this section, we present the different variants using two pheromone matrices.

F_i =Modify F(λ_i , F)

The two objective functions are simply aggregated with the factor λ_i $F = \lambda.f_1 + (1-\lambda_i).f_2$ where f_1 and f_2 are the two objective functions to optimize. This function F will be used when insert is applied in *GenerateSolutions* function.

F_i =Modify Tau(λ_i , F)

$\tau_{ij} = \lambda.\tau^1_{ij} + (1-\lambda).\tau^2_{ij}$ where τ^1_{ij} and τ^2_{ij} the pheromone matrices associated to f_1 and f_2, this pheromone matrix τ_{ij} will be used in the ACO algorithm for the construction of the sequence.

GenerateSolutions (F_i , T_i , s^*_{i-1})

This function is divided in three operations:

- construction of a sequence with an ACO algorithm
- local search:
    - insert
    - ND_LS
- update of the pheromone

For the update, two strategies are possible:

2pheroG

Here, we use a global strategy. Only the best solution $s^{1*}_i \in A$ according to f_1 is allowed to update the pheromone matrix τ^1_{ij} and only the best solution $s^{2*}_i \in A$ according to f_2 is allowed to update the pheromone matrix τ^2_{ij}. This means that the non dominated solution found since the beginning for all the λ vectors already examined, which has the best value for the makespan is allowed to update the pheromone matrix τ^1_{ij}. The same for the total tardiness and the pheromone matrix τ^2_{ij}.

2pheroL

The best solution $s^{1*}_i \in N^*$ according to f_1 is allowed to update the pheromone matrix τ^1_{ij} and solution $s^{2*}_i \in N^*$ according to f_2 is allowed to update the pheromone matrix τ^2_{ij}. This means that solutions which are allowed to update the pheromone matrices, are the non dominated solutions found for the current aggregation weight which are the best respectively for the makespan and the total tardiness .

UpdateArchive(N^*)

The purpose of this function is to add to the archive A all the solutions $s \in N^*$ which aren't dominated by any points $a \in A$. The second phase of this function is to filter A, this function deletes all dominated solutions in A and returns the filtered set.

s_i^*=SelectInitialSolution(N^*)

This function is applied when the variant *2phero 2phase* is chosen. As for the update of the pheromone, two strategies are possible:

- ***global***: the best solution $s_i^* \in A$ according to $F = \lambda.f_1 + (1-\lambda_i).f_2$ is chosen for the initialisation of the next aggregation *GenerateSolutions* function.

```
With the same notations than in Figure 21
Procedure 2pheroG
        for($i = 1,...,n$)
                $F_i$ =Modify F( $\lambda_i$, $F$ )
                $T_i$ =ModifyTau( $\lambda_i$, $T$ )
                $N^*$ =GenerateSolutions ( $F_i$, $T_i$, $s_{i-1}^*$ ,$A$)
                        Construction( $T_i$ )
                        Local search( $F_i$ )
                        Update ( $T_i$, $F$ ,$A$)
                end GenerateSolutions
                              $A$ =UpdateArchive( $N^*$ )
 (optional( $s_i^*$ =SelectInitialSolution( $A$ )) optional
        end for
end 2pheroG
```

Figure 22: Procedure of *2pheroG* algorithm

```
With the same notations than in Figure 21
Procedure 2pheroL
    for( $i = 1,...,n$ )
        $F_i$ =Modify F( $\lambda_i$, $F$ )
        $T_i$ =ModifyTau( $\lambda_i$, $T$ )
        $N^*$ =GenerateSolutions ( $F_i$, $T_i$, $s^*_{i-1}$ )
            Construction( $T_i$ )
            Local search( $F_i$ )
            Update ( $T_i$, $F$ , $N^*$ )
        End GenerateSolutions
        $A$ =UpdateArchive( $N^*$ )
 (optional( $s^*_i$ =SelectInitialSolution( $N^*$ )) optional
    end for
end 2pheroL
```

Figure 23: Procedure of *2pjeroL* algorithm

- ***local***: it is the best solution $s_i^* \in N^*$ which is chosen for the initialisation of the next step.

In Figures 22 and 23, we present the procedure of these two variants.

The choice of one strategy or the other probably depends on the region of the Pareto front which interests the decision maker. Intuitively, we assume that a *global* strategy for updating and initialisation should give better results at the extremes of the Pareto front whereas *local* strategy should be better in the middle.

In the experimental section, we will test and compare the different configurations on four different instances.

4.5 Performance measures

The determination of the Pareto front of a multiobjective optimization problem is a field studied by a large number of researchers. Thousand methods have been developed. To identify most promising optimizers and they must be compared. But then the question is how to compare their performance. The notion of performance includes two aspects, the quality of the outcome and computational resources needed to generate the outcome. Concerning the second aspect, there is no difference between single and multiobjective optimization whereas there is with the quality aspect. In practice, the overall running time

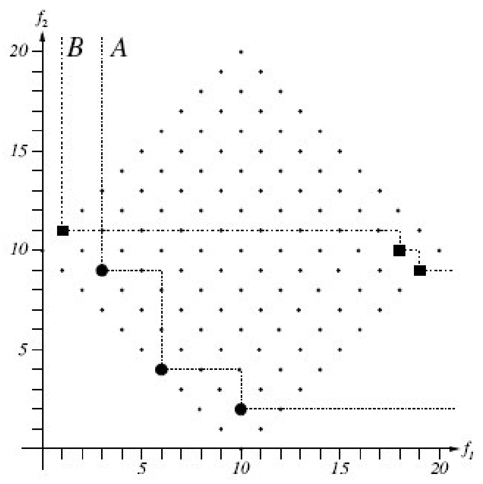


Figure 24: Limitations of a comparison based only on the dominance [93]

of each algorithm must be the same for all the runs and all the different optimizer tested and the difference of performance will be measured by the quality of the outcomes.

relation	notation	Interpretation in the objective space
Dominance or outperformance	$A \leq B$	Every $z^2 \in B$ is dominated by at least one $z^1 \in A$
Incomparable	$A \parallel B$	Neither $B \leq A$ nor $A \leq B$
equivalence	$A \sim B$	$A \leq B$ and $B \leq A$

Table 4 : Relation between two Pareto approximation sets

The quality of a solution for a MOP is something more difficult to define. In single objective problem, the quality can be defined by means of the objective function, the smaller (or the larger) the value, the better the solution. For multiobjective, the notion of quality is not clear anymore as different criteria can be used to define the quality of a solution. Another difference with the single objective problems is that in multiple objective problems, the outcome is not one point but a set of non-dominated points, what increases the difficulty of a comparison between two different algorithms.

We will call approximation set the set of non dominated points which results from one run of a multiobjective optimizer. If we consider two Pareto set approximations A and B, different relations exist between the two approximations sets (see Table 4)[159]

These three relations can be used to compare two optimizers, but a comparison based only on the notion of outperformance is very limited in practice. Actually even when two optimizers are incomparable, it is often possible to have a preference for one of them. In Figure 24, the two approximation sets are incomparable but a decision maker will in most situations prefer A.

The dominance relation also called outperformance is the simplest way to find that one algorithm is better than another but as this relation is imitated, another approach must be used.

In the literature, two different approaches are recommended [93]:

- unary quality measurement: a quality indicator (unary or binary) assigns each approximation set a measure which reflects a certain quality aspect or a combination of different quality aspects.
- attainment unction approach: an estimation of the probability of attaining arbitrary goals in the objective space.

In the following we will preset these two approaches and the unary indicator I_H, the hypervolume will be presented more in details.

4.5.1 Quality indicator

The idea of a quality indicator is to quantify differences between approximation sets by a real number. More formally,

An m-ary quality indicator I is a function $I : \Omega^m \rightarrow \Re$, which assigns each vector $(A_1, A_2, ..., A_m)$ of m approximation sets a real value $I(A_1, A_2, ..., A_m)$

In the literature, lots of unary and binary indicators can be found. All of them have their advantages and also their disadvantages like a loss of information occurred when the information are reduced to one number.

We will now present most interesting indicators

Unary indicator

A unary indicator associate a real value to an approximation set. Thus if A and B are two approximation sets, $I(A)$ and $I(B)$, their indicator value reveals a difference in the quality of the two sets. This indicator is commonly used, its capability of assigning quality values to approximation sets without considering other approximation sets makes it very attractive. Unfortunately this measure does not take into account the notion of dominance and Zitzler et al. have shown in [159] that generally unary indicators are not capable of indicating whether an approximation set is better than another. Nevertheless it exists unary

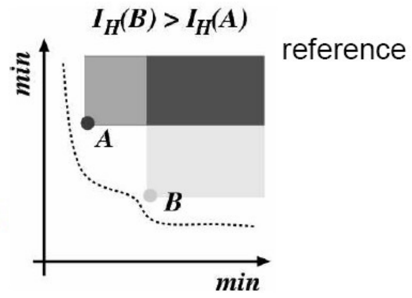


Figure 25: Illustration of an unary indicator: hypervolume [93]

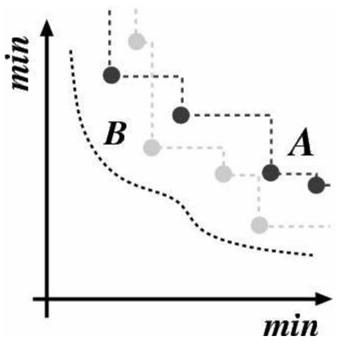
indicators which allow at best to infer that an approximation set is not worse than another. One of this unary indicator with this property is the hypervolume indicator I_H presented by Zitzler and Thiele in 1999[158]. The hypervolume is illustrated in Figure 25.

Let $X = (x_1, x_2, ..., x_l)$ be a set of decision vectors, $I_H(X)$ gives the volume enclosed by the union of the polytopes $p_1, p_2, ..., p_l$, where each p_i is formed By the intersections of the following hyperplanes arising out of x_i along with the axis: for each axis in the objective space, there exists a hyperplane perpendicular to the axis and passing through the point $\left(f_1(x_i), f_2(x_i), ..., f_k(x_i)\right)$. In two dimensions and for minimization problems, each p_i represents the rectangle defined by the points $\left(f_1(x_i), f_2(x_i), ..., f_k(x_i)\right)$ and $\left(f_1^{ref}, f_2^{ref}, ..., f_k^{ref}\right)$, where $\left(f_1^{ref}, f_2^{ref}, ..., f_k^{ref}\right)$ are the coordinates of a reference point which is dominated by all the decision vectors.

In Figure 24, we can observe that the hypervolume delimited by B is larger than the one by A. I_H has a desirable property that if $I_H(A) < I_H(B)$ and if all the points of A and B strictly dominate the bounding point, then A cannot be better than B. This desirable property makes the hypervolume a very interesting unary indicator [60].

Lots of other unary indicators such as the epsilon indicator I_ε^1 exist but we will not present them here. Anyway it must be noticed that each indicator exploits specific information and that they can give different results for the same comparison of two approximation sets. Hence, one approximation set A can be said to be better than an approximation set B only according to a specific indicator.

It is also important to notice that some indicators found in the literature are not Pareto compliant, what means that they give for result $I(A) > I(B)$ whereas $B > A$ if we consider Pareto dominance. I_H and I_ε^1 do not have this problem and it is one of the



$$I_C(A,B) = 0,25$$
$$I_C(B,A) = 0,75$$

Figure 26: Illustration of a binary indicator: coverage [93]

reasons why they are recommended for the comparison of two or more multiobjective optimizers [93]

Binary indicator

By contrast with unary indicators, it is only possible to compare two optimizers. Of course if we want to compare l optimizer, we will have to compute $l \cdot (l-1)$ tests instead of l tests for the unary indicator which is time consuming if a large number of algorithms must be compared. The advantage of binary indicator is that with some of them, it is sometimes possible to determine if one optimizer is better than another, if two optimizers are comparable or if they are equivalent. We will now present two binary indicators which have this property.

Illustrated in Figure 26, the coverage measure of two sets of decision vectors A and B is a function I_C which maps the ordered pair (A,B) to the interval $[0,1]$:

	A ≥ B	A = B	A ‖ B
I_C coverage	$I_C(A,B) = 1$ If $I_C(B,A) < 1$, then $A > B$	$I_C(A,B) = 1$ $I_C(B,A) = 1$	$I_C(A,B) < 1$ $I_C(B,A) < 1$
I_{H2} binary hypervolume	$I_{H2}(A,B) \geq 0$ $I_{H2}(B,A) = 0$	$I_{H2}(A,B) = 0$ $I_{H2}(B,A) = 0$	$I_{H2}(A,B) > 0$ $I_{H2}(B,A) > 0$

Table 5: Comparison based on binary indicator

$I_C(A,B) := \dfrac{|\{b \in B \mid \exists\, a \in A : a \geq b\}|}{|B|}$, thus $I_C(A,B) = 1$ means that all decision vectors which belong to B are dominated by at least one decision vector from A.

a binary version I_{H2} of the hypervolume I_H, also exists $I_{H2}(A,B)$ is defined as the hypervolume of the subspace that is weakly dominated by A but not by B.

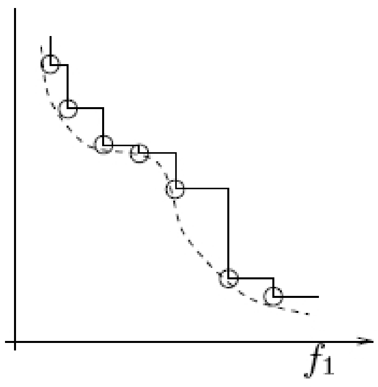


Figure 27: Plot of the attainment surface[60]

If $I_{H2}(A,B)=0$, then $B \geq A$. In Table 5, a summary of a comparison based on two binary indicators is presented.

The four quality indicators described will be used for the comparison of different approximation sets in the experimental part of the work.

4.5.2 Attainment functions

The second approach to compare multiobjective optimizers is the attainment function. The output of a single run of a multiobjective optimizer is an approximation set. This set is constituted by a certain number of non dominated solution vectors which can be plot on a graph. It would be possible to interpolate the points by a smooth curve, this curve represents an approximation of the Pareto front for this particular run of the optimizer. Actually this method is not safe and does not allow correct interpretation. Instead of being interpolated, these points can be replaced by a boundary. This boundary separates the points that are dominated by or equal to at least

one of the data points, from those that no data point dominates or equals. This boundary is called an attainment surface (see Figure 27).

This boundary is "the family of tightest goals known to be attainable as a result of the optimization run" [59].

Actually the attainment function provides a description of the distribution of an outcome set $X=(x_1,x_2,...,x_l)$ in a simple and elegant way, using the notion of goal-

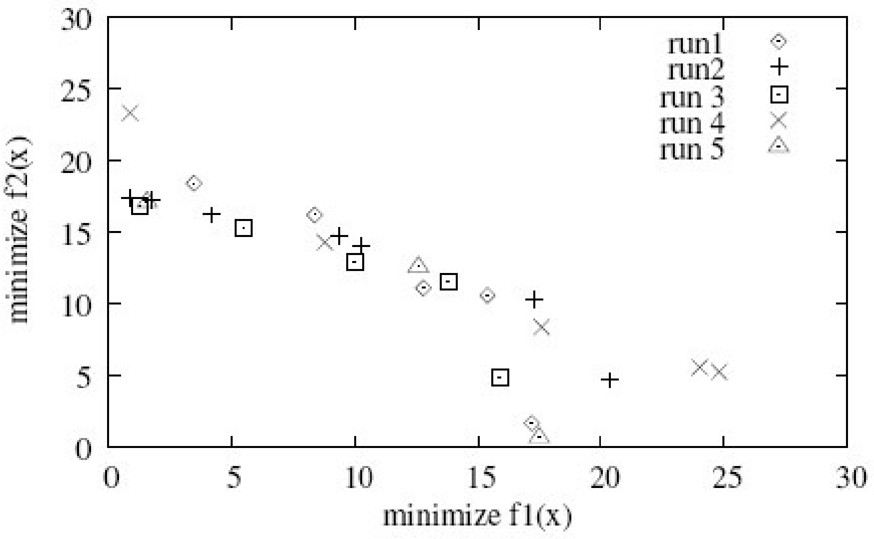


Figure 28: Superposition of 5 sets of non dominated points [93]

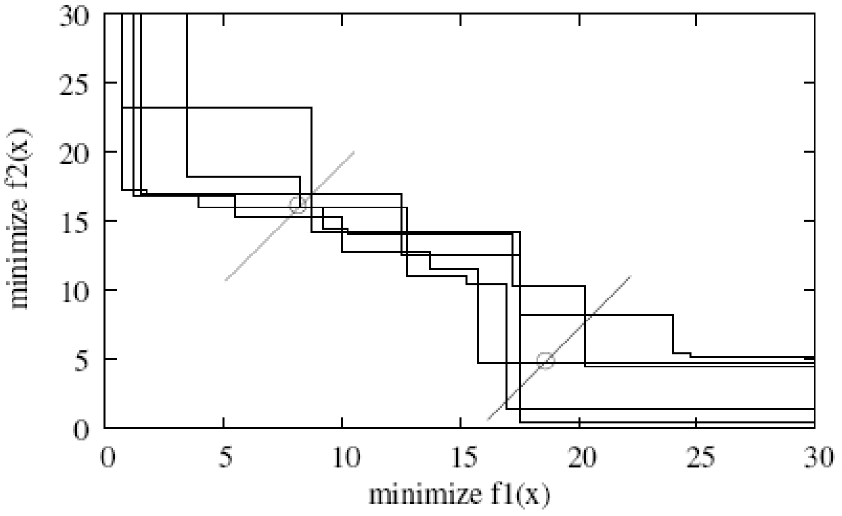


Figure 29: Superposition of the 5 corresponding attainment function [93]

attainment. It is defined by the function $\alpha_X : \Re^k \to [0,1]$ with $\alpha_X(z) = P(x_1 \le z \vee x_2 \le z \vee ... \vee x_l \le z) = P(X \le z)$.

It corresponds to the probability of at least one element of X being smaller than or equal to $z \in \Re^k$.

When we are faced to the execution of multiple runs of an optimizer, the display of the outcomes become rapidly confusing and misleading as shown in Figure 28. An interpretation of the results is very difficult and sometimes can lead to false conclusions.

plotting the corresponding attainment surfaces ‘see Figure 29) provides two clear information, the plot of multiple attainment surfaces also split the objective space in three , the region up and right is the region dominated by all the points obtained in the

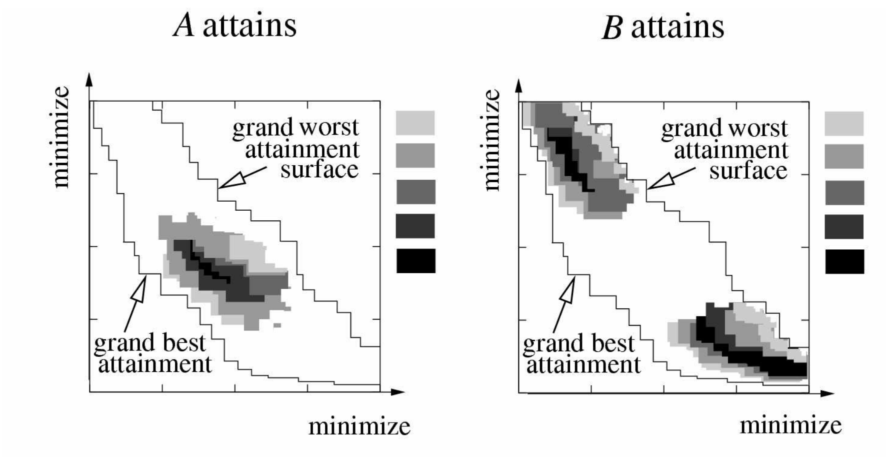


Figure 30: Plot of the difference of two empirical attainment functions [159]

approximation sets. Thus it is possible to visualize the worst case performance. By contrast the region left and down is the part of the objective space which is never attained by any points of the approximation sets. The best case can also be visualized. In between these two boundaries is a region that represents what has been attained in some runs but not in others.

In practice, the attainment function can be estimated via its empirical counterpart: Let $X_1, X_2, ..., X_n$ be the random set which corresponds to n independent runs of the optimizer, the empirical attainment function is defined by $\alpha_n(z) = \frac{1}{n}\sum_{i=1}^{n} I\{X_i \leq z\}$ where $I()$ is the indicator function which evaluates to one if its argument is true and 0 otherwise. Thus the empirical attainment function gives for each objective vector in the objective space the relative frequency that it was attained, weakly dominated by an approximation set, with respect to the n runs. This function will be very useful for the visualization of different runs of different optimizers (see Figure 30) and can be seen as a distribution of the solution quality.

Before investigating the differences in the attainment of two optimizer, it is recommended in [93] to run a Kolmogorov-Smirnov test[3] (K-S test) to probe the difference in the empirical attainment functions of two optimizer A and B. The null hypothesis is that the attainment surfaces A and B are identical and the alternative hypothesis is that the

[3] In statistics, the Kolmogorov–Smirnov test (often called the K-S test) is used to determine whether two underlying one-dimensional probability distributions differ, or whether an underlying probability distribution differs from a hypothesized distribution, in either case based on finite samples[143]

distributions differ somewhere. If the null hypothesis is rejected, investigate the differences in the two attainment functions becomes possible.

Manuel Lopez Ibanez has proposed in [99] an elegant method to visualize the difference between two approximation sets which have been run several times

The method as presented in [93] is the following:

- Concatenate all runs from A and B and compute the grand best and grand worst attainment surfaces. Grand worst and grand best surfaces are respectively the line which connects the set of points attained by any run of the two configurations and the line which connects the best set of point attained all over the run
- Compute all goals where there is a statistically significant difference in probability of attaining that region between algorithm A and B.
- Either: plot the difference in empirical frequency of attaining those goals where A is better in the left plot (and where B is better in the right plot).

This method provides a clear visualization of locations where an algorithm performs better than another and how much. If we look at Figure 29, we can see that A is better than B in the middle and that B is better at the extreme. The different shades of greys give the differences between attainment functions, the darker the points, the larger the difference.

5

Experiments

Considering the experimental section of the work, it will be divided in two main parts. We have in a first time considered a single objective approach where the purpose is to compare the performance of the different configurations of algorithms for each objective separately. These experiments will provide information which will be taken into account for the design of the two proposed biobjective algorithms.

. In the second part, we will present the details of the different ACO approaches we proposed to tackle the flowshop problem and we will describe our experiments.

Finally we will present and discuss the results provided by our different tests and comparisons

Above all we begin with the description of the ACO algorithms used for the different tests. The global procedure of the algorithm is the following:

```
procedure Ant Colony Optimization
  Initialize the parameters
  while (end condition not met) do
    construct a solution
    apply a local search
    update the pheromone trails
  end
end Ant Colony Optimization
```

Figure 31: Procedure of the single objective ACO algorithm

We have implemented two different versions of ACO algorithms for the construction of the solutions:

– Max Min Ant System (MMAS see section 3.3.5).

- A variant of MMAS incorporating the summation rule introduced by Merkle and Middendorf in 2000 [105]; we call it MMAS_sum.

These algorithms have been associated with different combinations of the three local search neighbourhood algorithms presented in section 3.2.2:

- trans
- exchange
- insert

Then we have implemented a multiobjective approach which takes into account the results found during the single objective study. In that second section we have also studied different configurations and variants of our algorithms. Before presenting and commenting the results of these experiments, we will first present in detail the two ACO algorithms and the different parameters which are part of the procedure such as the aggregation strategy.

5.1 Description of the ACO algorithms

As said before, in a PFSP with n jobs, the problem is to find a permutation of n $\{1,...,n\}$ which minimizes the objective function. In this context, the quantity of pheromone τ_{ij} represents the desirability of setting a job i at the j^{th} position in the sequence. in scheduling problems, very often an ant constructs a sequence by first choosing a job for the first position, then a job for the second position and so on until all the jobs are scheduled. The way an ant chooses a job i to set at position j depends on a probability p_0. It makes with a probability p_0 the "best decision" and chooses with a probability $1-p_0$, a job according to a pseudo random proportional rule.

5.1.1 Max Min Ant System

In MMAS, the "best decision" means choosing the job i with the maximal value of τ_{ij} that is, the job i with the maximal desirability for position j. Whereas with a probability $1-p_0$, the ant chooses the job according to the following distribution of probability:

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}}{\sum_{i \text{ not scheduled}} \tau_{ij}} & \text{if job } i \text{ is not yet scheduled} \\ 0 \text{ otherwise} & \end{cases}$$

In MMAS as in ACS, only one ant, the ant having found the best solution, is allowed to update the pheromone matrix according to the following formula: $\tau_{ij}^{new} = (1-\rho).\tau_{ij}^{old} + \Delta\tau_{ij}$ were ρ, with $0 < \rho < 1$, is the evaporation rate.

Another characteristic of MMAS is that the pheromone τ_{ij} belongs to the interval $[\tau_{\min}, \tau_{\max}]$.

5.1.2 Max Min Ant System incorporating the summation rule

This algorithm differs a little bit from MMAS in the construction phase of the solution. Instead of exploiting the pheromone τ_{ij}, it uses a new parameter $T_{ij} = \sum_{q=1}^{j} \tau_{iq}$ in the decision process of choosing a job i to be set at position j. Then, among the first five unscheduled jobs, with a probability p_0, the job i with the maximal value T_{ij} is chosen for position j in the best sequence obtained so far. The pseudo random proportional choice is applied with a probability $1-p_0$ and the job i among the five first unscheduled jobs is selected according to the following probability distribution:

$$p_{ij} = \begin{cases} p_{ij} = \dfrac{T_{ij}}{\sum_l T_{lj}} & \text{if job } i \text{ is not yet scheduled} \\ 0 \text{ otherwise} & \end{cases}$$

where l belongs to the set of five unscheduled jobs. If there are less than five jobs unscheduled, all the jobs are considered.

5.2 Local search

We have combined these three local search algorithms (trans, exchange and insert) in order to improve their performance. For example, we know that transpose neighbourhood requires few computational time and quickly gives an improved solution. Thus we have tried to use it before applying a local search algorithm searching the insert neighbourhood which typically requires more time to find an improved solution in the beginning of the process. We have decided to try seven different combinations of local search:

- Insert (I)
- trans followed by insert (T-I)

- trans followed by exchange (T-E)
- exchange followed by insert (E-I)
- insert followed by exchan(I-E)
- Trans followed by insert and then by exchange (T-I-E)
- Trans followed by exchange then by insert (T-E-I)

5.3 Single objective approach

5.3.1 Instances and parameters

We have first attacked the permutation flowshop problem with the makespan objective under consideration. We have tested the different algorithms on five instances taken from the well known Taillard benchmark. The instances of Taillard form a set of 120 problems of various sizes, having 20, 50, 100, 200 and 500 jobs and 5, 10 or 20 machines. These problems are extremely difficult to solve and are a good benchmark to test the different methods of optimization. These five chosen instances are presented in Table 6.

instances	number of jobs n	number of machines m
tai51	50	20
tai61	100	5
tai71	100	10
tai81	100	20
tai91	200	10

Table 6: Taillard instances used for the single objective test

Then we have used some of the instances proposed by Ruiz on his website[4] to test the performance of our algorithms for the problem with the total tardiness objective. The two instances chosen are presented in Table 7.

instances	number of jobs n	number of machines m
I_0,2_0,2_50_10_1	50	10
I_0,2_0,2_50_50_1	50	50

Table 7: Eva instances used for the single objective test

We have tested different configurations by combining the two different ACO algorithms with the seven different combinations of local search and seven values of p_0, $p_0 = \{\text{¨}0, 0.5, 0.75, 0.9, 0.925, 0.95, 0.975\}$.

[4] http://www.upv.es/gio/rruiz/

We have also decided to fix the update parameters to common values: $\rho = 0.2$, $\tau_{\min} = 1$, $\tau_{\max} = \frac{\tau_{\min}}{\rho}$, $\tau_{ij}(0) = \tau_{\max}$.

5.3.2 Results

In order to make fair comparisons, the different algorithms have been run independently ten times for each instance, using as stopping criterion the same maximum computation time.

For all these instances frequently used to test the performance of algorithms, a best solution found so far is given. Hence we have used it to calculate the relative error of each algorithm for each instance and we have compared these values to determine which configuration performs the best. The performance measure is $error_{rel} = \frac{a\lg o_{sol} - best_{sol}}{best_{sol}}$, the percentage increase over the optimum, where $a\lg o_{sol}$ is the average of the ten solutions given by the tested algorithm and $best_{sol}$ the lowest known upper bound.

We first present the results of the comparison with the makespan for objective and then for the total tardiness. In Table 8, we present the averages of the relative error for the five instances for the different combinations of local search algorithms, for different values of p_0 for the two ACO algorithms. We can observe that the best results (in bold) are obtained for a high value of p_0 for both ACO algorithms and that the best configurations for the local search are I and T-I. The details of this comparison are presented in appendix 1. They show that for each instance, best results are obtained with high value of p_0 and with insert or Trans-Insert for local search. This result is due to the speedup technique of Taillard which can be used for Insert when the makespan is the objective to optimize but not for the total tardiness. We can also observe that MMAS_SUM performs better than MMAS.

	MMAS							MMAS_SUM						
p0	*I*	*T-I*	*T-E*	*I-E*	*E-I*	*T-I-E*	*T-E-I*	*I*	*T-I*	*T-E*	*I-E*	*E-I*	*T-I-E*	*T-E-I*
0	2.4%	2.4%	3.8%	2.5%	2.6%	2.5%	2.6%	2.0%	2.0%	3.4%	2.3%	2.4%	2.3%	2.7%
0.5	2.3%	2.6%	3.8%	2.5%	2.6%	2.5%	2.6%	1.9%	1.9%	3.3%	2.2%	2.4%	2.2%	2.3%
0.75	2.1%	2.2%	3.6%	2.7%	2.5%	2.7%	2.5%	1.8%	1.8%	3.4%	2.4%	2.3%	2.4%	2.3%
0.9	1.9%	1.9%	3.4%	2.3%	2.4%	2.2%	2.4%	1.7%	1.7%	3.1%	2.1%	2.2%	2.1%	2.3%
0.925	1.8%	1.9%	3.4%	2.2%	2.4%	2.1%	2.4%	**1.6%**	**1.6%**	3.4%	2.0%	2.3%	2.1%	2.2%
0.95	**1.7%**	1.8%	3.5%	2.1%	2.3%	2.1%	2.2%	1.7%	1.7%	3.5%	2.0%	2.2%	2.1%	2.2%
0.975	1.8%	**1.7%**	3.4%	2.1%	2.2%	10.7%	11.3%	1.7%	1.7%	3.6%	2.1%	2.3%	2.1%	2.3%

Table 8: Summary of the relative errors for the makespan

In Table 9, we present the averages of the relative error for the two instances for the different combinations of local search algorithms for different values of p_0 for the two ACO algorithms. We can observe that best the best results (in bold) are obtained for a high value of p_0 for both ACO algorithms and that. For the local search, none of them seems to perform better than the others. The only conclusion we can make is that it should include insert. The details of this comparison are presented in appendix 3.

	MMAS							MMAS_SUM						
P0	I	T-I	T-E	I-E	E-I	T-E-I	T-I-E	I	T-I	T-E	I-E	E-I	T-E-I	T-I-E
0	13.4%	12.7%	18.3%	14.2%	12.7%	12.9%	12.9%	8.5%	7.8%	12.9%	9.0%	7.3%	7.2%	7.7%
0.5	11.9%	11.3%	17.1%	12.5%	12.3%	10.6%	10.8%	6.6%	6.8%	11.0%	7.6%	6.5%	6.3%	6.6%
0.75	11.9%	11.3%	17.1%	12.5%	12.3%	10.6%	10.8%	6.5%	4.6%	9.5%	6.4%	5.3%	5.3%	5.3%
0.9	9.6%	7.1%	11.2%	9.4%	8.6%	6.5%	6.2%	6.2%	5.2%	7.7%	6.8%	6.2%	6.0%	5.5%
0.925	10.3%	6.1%	13.2%	10.3%	7.0%	7.4%	**5.7%**	6.0%	4.5%	8.3%	6.4%	**4.3%**	5.8%	5.2%
0.95	10.5%	5.8%	14.1%	10.8%	7.4%	7.7%	7.1%	5.2%	4.6%	9.4%	7.5%	5.8%	5.9%	6.9%
0.975	9.9%	7.9%	14.0%	11.6%	8.7%	8.8%	9.5%	8.8%	9.1%	9.9%	9.8%	8.1%	8.4%	8.8%

Table 9 : Summary of the relative errors for the total tardiness

5.4 Biobjective approach

We have already presented the two biobjective ACO algorithms in section 4.4, in this section we will just detail the function *Generate Solution* explicitly for the two approaches. The single objective tests have provided three interesting information:

- The use of MMAS_sum for the ACO algorithm gives better results
- Insert neighbourhood must be include in the local search.
- A high value of p_0 must be chosen.

For the choice of the local search, we have made several comparisons based on quality indicators (see appendix 3). The outcome of this comparison is that no relation of outperformance, neither significant difference in the unary and binary hypervolume I_H and

I_{H2} can be observed. Thus for the following experiments, we have decided to use insert combined with the non dominated local search (ND_LS)introduced in section 4.4.1.

5.4.1 ACO algorithm using one pheromone matrix (1phero)

GenerateSolutions ($\mathbf{F_i . T_i . s^*_{i-1}}$)

This function is divided in three phases:

- construction of a sequence
- local search
    - insert
    - ND_LS
- update of the pheromone

For this function, we have used a combination of MMAS_sum for the construction of the sequence with two local search, Insert with $F = \lambda.f_1 + (1-\lambda_i).f_2$ for objective and ND_LS. The final result of the resulting algorithm is denoted by N^*.

Insert is necessary if we want to reach good quality solutions as it is known for flowshop scheduling problem, that an ACO algorithm used without local search is not well performing.

For the pheromone update, only the best solution $s_i^* \in N^*$ according to $F = \lambda.f_1 + (1-\lambda_i).f_2$ is allowed to update the pheromone matrix with this following rule:

$\tau_{ij}^{new} = (1-\rho).\tau_{ij}^{old} + \Delta\tau_{ij}$ with $\tau_{\min} \le \tau_{ij} \le \tau_{\max}$ and $\Delta\tau_{ij} = \begin{cases} 1 \text{ if } i \text{ is set at the position } j \\ 0 \text{ otherwise} \end{cases}$. In our algorithms, we have chosen to have a constant quantity of pheromone deposited at each update.

5.4.2 ACO algorithm using two pheromone matrices (2phero)

We have already presented this algorithm and its variants in section 4.4. We just precise that MMAS_sum is the ACO algorithm used for the construction of the solution

For the local search, we use insert following by ND_LS. The pheromone matrices are updated according to the same rule than *1phero*, the difference appears only in the solutions which are allowed to update the pheromone

In the following, we will test the performance of four configurations of algorithms with two pheromone matrices:

- *2pheroG scratch*
- *2pheroG 2phase*
- *2pheroL scratch*
- *2pheroL 2phase*

5.4.3 Instances and parameters

Performance of the different algorithms has been tested on four different instances provided by Ruiz on his website. The chosen instances differ in the number of machines and the value of the linear correlation between the two objectives. We assume that these two parameters have an impact on the performance of the different configurations and we will try to have an idea on their influence.

For each instance, we have calculated the linear correlation coefficient by generating randomly ten thousand sequences of n jobs and calculating their makespan and total tardiness values, respectively.

instance	name	number of jobs n	number of machines m	correlation coefficient
50x10-1	I_0.2_0.2_50_10_1	50	10	0.57
50x10-2	I_0.6_0.2_50_10_1	50	10	0.40
50x30-1	I_0.2_0.2_50_30_1	50	30	0.34
50x30-2	I_0.2_I_50_30_1	50	30	0.25

Table 10: Eva instances used for the biobjective tests

For the parameters, we have chosen common values used in MMAS algorithms:

$$p_0 = 0.95\,,\ \rho = 0.2\,,\ \tau_{\min} = 1,\ \tau_{\max} = \frac{\tau_{\min}}{\rho}.$$

5.4.4 Comparison procedure

The comparison procedure of the different optimizers A and B is the following:

1. Run each configuration ten times on each instance. for the same computation time
2. check the outperformance relation between the two optimizers by calculating the coverage $I_C(a,b)$ and $I_C(b,a)$ for each possible pairs (a,b) where a is one

approximation set obtained by one run of A and b one approximation set obtained by one run of B. Thus we have 100 different values of $I_C(a,b)$ and $I_C(b,a)$.

- $I_C(b,a)=1$ and $I_C(a,b)<1 \forall a \in A, \forall b \in B$, B outperforms A and vice versa.
- $I_C(a,b)\approx 1$ and $I_C(b,a)\approx 1$ $\forall a \in A, \forall b \in B$, $A=B$
- the average of $I_C(a,b)$ is clearly superior to $I_C(b,a)$, we have an indication that A is better than B and vice versa
- $I_C(a,b)\approx 1$ and $I_C(b,a)\neq I_C(a,b) \forall a \in A, \forall b \in B$ but no clear superiority are observed, A and B are probably incomparable.

3. Calculate the respective empirical attainment function (EAF).
4. Perform the Kolmogorov-Smirnov test with the null hypothesis that the two empirical attainment surfaces are identical. The alternative hypothesis is that they differ somewhere.
5. If the null hypothesis is rejected, investigate the differences in the two EAFs with the visualization proposed by Manuel López-Ibáñez in [99].
6. if necessary, calculate the unary hypervolume I_H average over the ten runs for each configuration and the binary hypervolume $I_{H2}(a,b)$ for each possible pairs (a,b), where a is one approximation set obtained by one run of A and b one approximation set obtained by one run of B. These indicators will be used to provide more information on the quality of the approximation sets.

This procedure will be used for most of the following comparisons.

We will now present an example were we compare *2pheorG scratch* and *2pheroL scratch* on 50x30-1.

Table 11 shows the values of the coverage measure and the result of the Kolmogorov Smirnov test for a comparison between the configurations *2heroG scratch* and *2pherLG scratch* for a direction changes $\lambda = 1 \square\ 0$, which means that the weight begins with $\lambda = 1$ and that the makespan is the most important objective at the beginning..

The tests are run on instance 50x30-1 with 50 jobs and 30 machines.

λ = 1□ 0 - 50x30-1	I_C	Kolmogorov-Smirnov test
(2pheroG scratch/2pheroL scratch)	39%	The two attainment surfaces differ somewhere
(2pheroL scratch/2pheroG scratch)	52%	

Table 11: Results of a comparison *global/local* strategy (50x30-1)

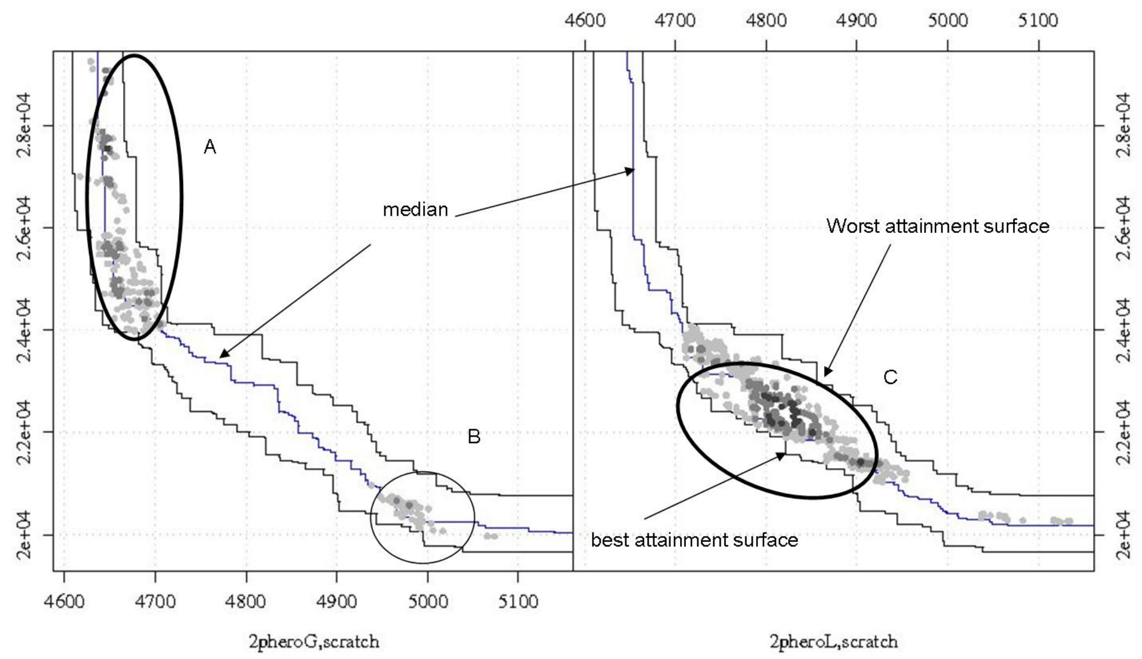


Figure 32: Differences of EAFs, *global scratch/local scratch / 2pheroG 2phase* (50x30-1)

■ [1.0,0.8[
▩ [0.8,0.6[
▨ [0.6,0.4[
□ [0.4,0.2[

Figure 33: Grey scale encoding of the difference EAFs

The coverage measure does not provide outperformance relation, neither clear preference. The result of the second line of Table 11 means that 39% of the solutions obtained by *2pheroL scratch* are dominated by at least one solution obtained by *2pheroG scratch*. The second line indicates that 52% of the solutions obtained by *2pheroL scratch* are dominated by at least one solution obtained by *2pheroG scratch*. It can be noticed that relations of outperformance are rare and that most of time, comparisons will be based on the plot of the differences of two EAFs if the Kolmogorov-Smirnov, test of equality of two EAFs is rejected, we can visualize the differences of two EAFs which can provide information on where and how large are the differences. The size of the difference between the two EAFs is represented by different shades of grey. Darker points represent larger difference. The difference scale is given in Figure 33.

In Figure 32 and for all the next plot of the differences of two EAFs, only differences above 0.2 are plot. The sign of the difference gives information about which configuration performs better than the other at one point; the name at the bottom of each plot indicates for which of the two algorithms the differences shown are positive. Points in the left plot show the region where 2*pheroG scratch (A and B)* is better whereas points in the right plot show the region where *2pheroL* scratch performs better (C). We can observe that *2pheroG scratch* performs better than 2*pheroL scratch* in the extreme whereas *2pheroL* scratch is more performing in the middle.
Three other information are provided by the plot. On both plots, the upper line connects the set of points attained by any run of the two configurations and the lower line connects the best set of point attained all over the runs. The line between the two represents the median for each algorithm. For this plot and the remaining plots, the x-coordinate is the makespan and the y-coordinate the total tardiness.

5.4.5 Aggregation strategy

The aggregation strategy is a very important factor which influences the performance of the algorithms and their different variants. In order to have a better idea of its influence, we have compared three different numbers of aggregations. We have also compared the results obtained by three configurations for different direction changes.

Influence of the number of weights

For the number of weights, a trade-off must be found between the number of aggregation weights and the time spent to search for solutions for each different weight. If we fix the computation time of the whole procedure, a larger number of weights implies that for each weight λ_i, the time spent to find non dominated solutions will be shorter. Nevertheless a large number of weights increases the possibility of being closer to the theoretical Pareto front. In fact if the difference between two consecutives weights is too large, the probability of missing non dominated solution in this region of the objective space increases. Furthermore, with a large number of weights, more non dominated solutions will be found.

We have tested three different numbers of weights on 50x10-2and 50x30-2 for three different configurations (1phero *2phase*, *2pheroG 2phase and 2pheroL 2phase).*

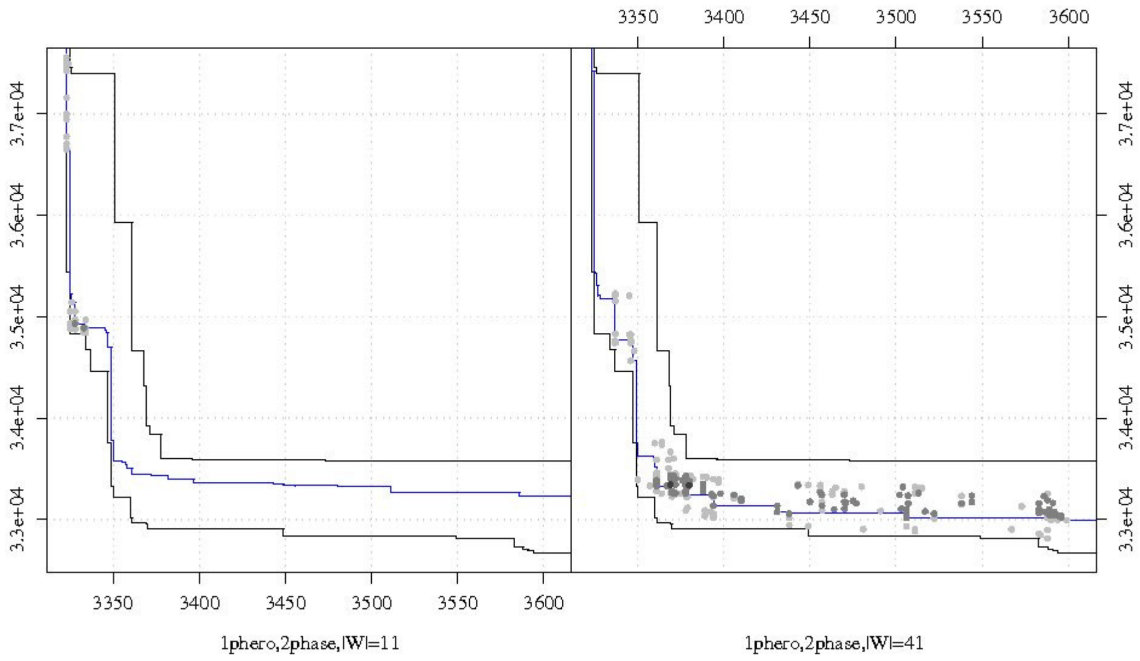


Figure 34 : Differences of EAFs, *1phero 2phase* |W|=11-|W|=41 (50x10-2)

As for all the different tests, we have run ten times each algorithm using for stopping criterion the same amount of computational time. We have used three different numbers of aggregation weights, $|W| \in \{11,41,81\}$, the details of these comparisons are presented in appendix 4.

The main observations provided by these tests on the instance with 50 jobs and 10 machines are:

- *1phero 2phase* with $|W| = 41$ or $|W| = 81$ is better than *1phero 2phase* with $|W| = 11$. This is illustrated in Figure 34.
- There are no significant difference between using $|W| = 41$ or $|W| = 81$.(see Table 12)

Table 12 presents the results obtained for the configuration *1phero 2phase* tested on t50x10-2 for a direction changes $\lambda = 1\square\ 0$ and Figure 34 presents the differences of EAFs between using $|W| = 11$ and $|W| = 41$ for the configuration *1phero 2phase*.

1phero 2phase - $\lambda = 1$□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
$(1phero\ 2phase - \lvert W \rvert = 11), (1phero\ 2phase - \lvert W \rvert = 41)$	25%	The two attainment surfaces differ somewhere
$(1phero\ 2phase - \lvert W \rvert = 41), (1phero\ 2phase - \lvert W \rvert = 11)$	65%	
$(1phero\ 2phase - \lvert W \rvert = 11), (1phero\ 2phase - \lvert W \rvert = 81)$	28%	The two attainment surfaces differ somewhere
$(1phero\ 2phase - \lvert W \rvert = 81), (1phero\ 2phase - \lvert W \rvert = 11)$	62%	
$(1phero\ 2phase - \lvert W \rvert = 41), (1phero\ 2phase - \lvert W \rvert = 81)$	49%	h_0 not rejected
$(1phero\ 2phase - \lvert W \rvert = 81), (1phero\ 2phase - \lvert W \rvert = 41)$	42%	

Table 12 : Results for different numbers of weights for *1phero 2phase* (50x10-2)

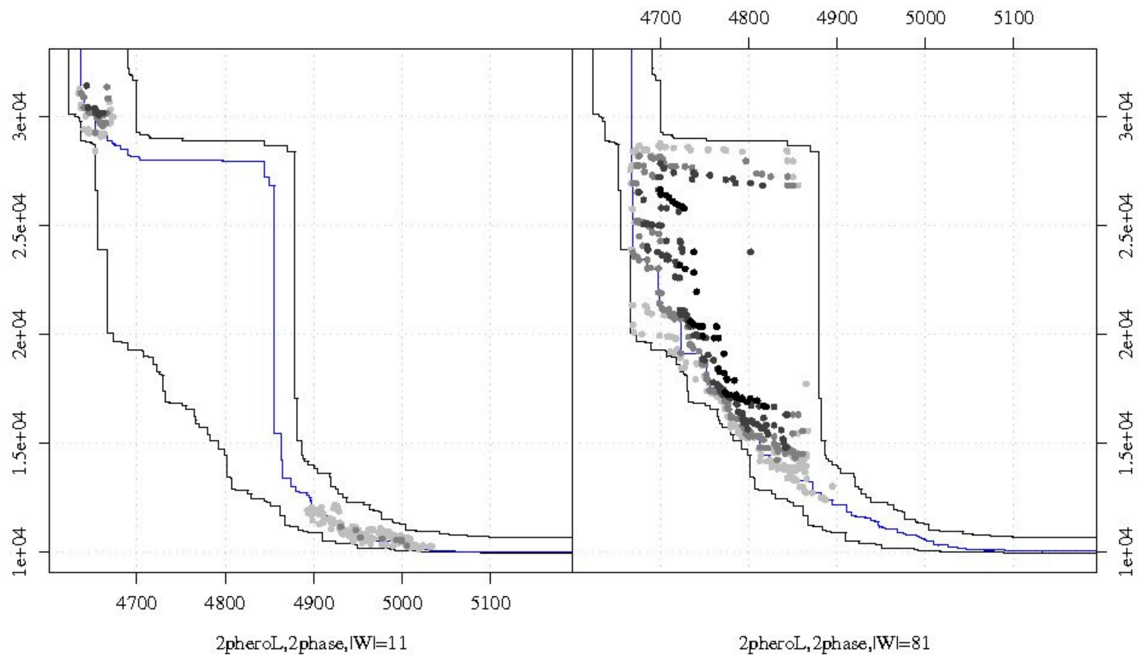


Figure 35 : Differences of EAFs, *1phero 2phase* |W|=11-|W|=81 (50x30-2)

- For *2pheroG 2phase* and *2pheroL 2phase*, the number of aggregation weights has not a clear influence on the performance (see Tables 22 and 23 in appendix 4). In this section, For the instance with 30 machines, the observations are slightly different:

- Using $\lvert W \rvert = 11$ seems to be better to find good solutions in the left upper region of the objective space where the makespan is more important. The use of $\lvert W \rvert = 41$ or $\lvert W \rvert = 81$ provides better results in the middle. Figures 35, 45 to 48 and 50 in appendix 4 clearly show the differences in the behaviour of the same algorithm for two different numbers of weights.

Figure 35 suggests that a certain number of weights is necessary to find solutions located in the middle of the objective space. The fact that better solutions are found in the extreme when using $|W| = 11$ is just the sign that better solutions can be found if more time is allocated to the whole procedure for $|W| = 41$ or $|W| = 81$.

Conclusion on the influence of the number of weights

- A minimum number of weights is necessary to find a good approximation of the Pareto front in the middle
- After a certain number of weights (probably problem dependent) increasing the number of weights does not induce improvement of the solution quality anymore

Influence of the direction changes

For a *scratch* approach, the way the aggregation weights vary is not important, as the algorithm starts from scratch for each weight, only the value of the weight is important and used in the algorithm. How this value has been calculated does not influence the procedure.

But for the *2phase* approach, several differences could appear following the direction changes. As the two objectives are different the results could also be different if the Pareto front is covered in one sense or another. Beginning a new phase with a solution of very good quality for one objective can induce problems to find good solutions for the second objective and thus problems in finding good non dominated solutions in the region of the objective space with low values for the second objective. Thus depending of the objectives, the instance and the algorithm used, a clever choice for the direction changes can be useful to reach better performances.

In section 4.4 we have defined the aggregated function $F = \lambda.f_1 + (1-\lambda_i).f_2$. In our algorithms, f_1 and f_2 are respectively the makespan and the total tardiness. For our tests, we have tried different direction changes during the search procedure for the configurations using a *2phase* approach:

– $\lambda = 1 \rightarrow 0$ with a gradual decrease, $\lambda_i = \lambda_{i-1} - \frac{1}{|W|-1}$, this means that at the beginning of the procedure the most important objective is the makespan. During all the process, makespan looses in importance and at the end, only the total tardiness is considered by the algorithm.

– $\lambda = 0\square\ 1$ with a gradual increase $\lambda_i = \lambda_{i-1} + \frac{1}{|W|-1}$, it is the opposite situation than the previous direction changes.

– $\lambda = 0\square\ 1\square\ 0$ with $\lambda_i = \lambda_{i-1} \pm \frac{1}{|W|-1}$, the procedure focuses first on solutions with best tardiness before covering the front in the other sense.

– $\lambda = 1\square\ 0\square\ 1$ with $\lambda_i = \lambda_{i-1} \mp \frac{1}{|W|-1}$, the opposite of the previous situation.

The two last direction changes are double 2phase approaches where the Pareto front is covered in one sense and then in the other. This approach has previously been tested in [133].

Double 2pase approach allows to avoid the problem described before.

We have tested different direction changes on 50x10-2 and 50x30-2 for three configurations: *1phero 2phase*, *2pheroG 2phase* and *2pheroL 2phase*. The details of these experiments are presented in appendix 5.

We present now the main observations of the tests on 50x10-2:

- For *1phero 2phase*, if it seems that using a direction changes beginning with $\lambda = 1$ helps to find solution with small makespan (see Figures 54 and 55 in appendix 5)
- For *2pheroG 2phase*, the double 2phase approach $\lambda = 1\square\ 0\square\ 1$ seems to be less performing than the two direction changes beginning with $\lambda = 0$ (see Figure 56, 57 in appendix 5)
- For *2pheroL 2phase*, choosing $\lambda = 1\square\ 0$ is worse than the three other possibilities tested. If we compare the two double 2phase approaches $\lambda = 0\square\ 1\square\ 0$ and $\lambda = 1\square\ 0\square\ 1$, we have an indication that the second evolution could give better results for low values of makespan while the other would give slightly better solutions for the middle and the right region of the objective space (see Figure 56 in appendix 5).
- They also suggest for *1phero 2phase* and *2pheroL 2phase* that using the double 2phase approach beginning with $\lambda = 1$ helps to find good solutions for non dominated solutions with a low value of makespan whereas using the other double phase approach

$\lambda = 0$□ 1□ 0 would slightly help to find good solutions in the middle and in the right region of the objective space.

More generally, the results of the comparisons suggest that a double 2phase approach would generally give slightly better results for 50x10-2. It is illustrated in Figure 36.

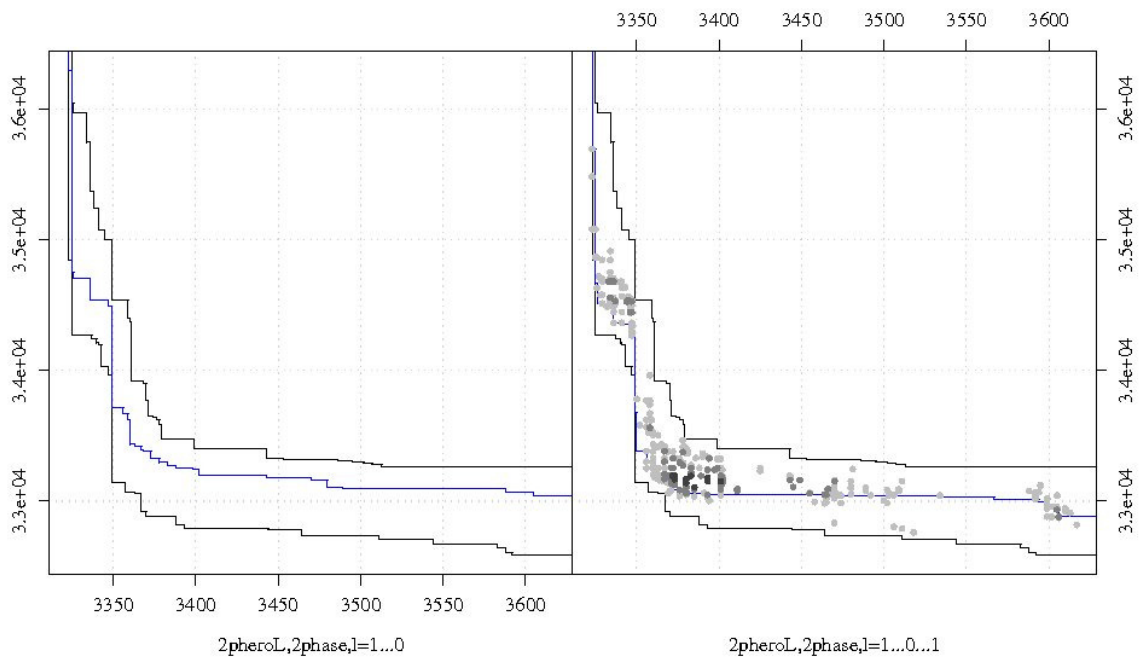


Figure 36 : Differences of EAFs, *2pheroL* 2phase for direction changes (50x10-2)

For 50x30-2, observations are different:

- For *1phero phase,* the direction changes $\lambda = 0$□ 1 seems to be more appropriated for this instance and for most of regions of the objective space (see Table 30 and Figures 59, 60 in appendix 5).
- For *2pheroG 2phase,* an evolutions beginning with $\lambda = 0$ are clearly better for the instance and for almost all regions of the objective space. Among the two, the direction changes $\lambda = 0$□ 1 seems to be slightly better (see Table 32 and Figure 62 in appendix 5).
- For *2pheroL 2phase*, best performances seems to be obtained when the direction changes is $\lambda = 1$□ 0□ 1.

Conclusion on the influence of the direction changes

- Performance of a configuration and type of direction changes are correlated even if it is difficult to say how.
- Following the direction changes chosen, the result of a comparison between two configurations can be opposite.
- For the 50x10 problem, direction changes beginning with $\lambda = 1$ could give better results for the region where the makespan is more important

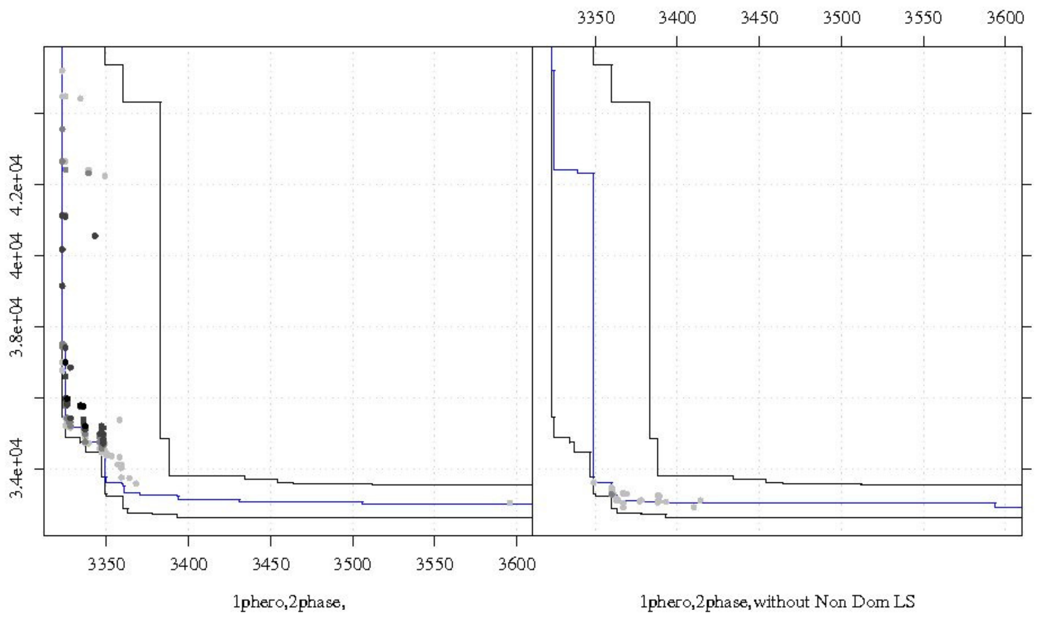


Figure 37: Differences of EAFs, *1phero 2phase* with/without ND_LS (50x10-2)

5.4. 6 Influence of the non dominated local search

In this section, we present the observations concerning the usefulness of ND_LS. We have compared three configurations using the non dominated local search and the three same configurations without D_LS. The three configurations *1phero 2phase*, *2pheroG 2phase* and *2pheroL 2phase* have been tested on t50x10-2 and 50x30-2 with $|W| = 41$ and $\lambda = 1$□ 0.

The results show that the configuration with ND_LS is never worst than the configuration without, moreover it appears that it can sometimes perform better as illustrated in Figure

37. The configuration with ND_LS is better for the left region of the objective space and is not worse than the configuration without ND_LS for the other regions. More results are presented in appendix 6.

Conclusion on the influence of ND_LS

- The use of non dominated local search (ND_LS) never makes things worse and can sometimes improve the performance

5.4. 7 Comparisons between the different configurations

In this section, we present the results of a series of computational experiments conducted to test and compare the effectiveness of the two ACO algorithms proposed and their different variants.

We have essentially focused on three points for the comparison of the different configurations. We want to know if possible:

1. whether it is more relevant to use one or two pheromone matrices in our ACO algorithm, in which region of the objective space and for which instance
2. whether scratch or *2phase* approach provides better results,
3. whether *local* or *global* strategy for *2phero* performs better than the other

In the following, we will present for each question the corresponding results obtained for the four different instances.

For the experiments, we have fixed the number of weights $|W| = 41$ and the direction changes $\lambda = 1\square\ 0$. Table 14 presents a summary of the observations obtained for the four instances. In this table we will use the following notations to describe the observations:

– 1_scr: *1phero scratch*
– 1_2ph: *1phero 2phase*
– 2G_scr: *2pheroG scratch*
– 2G_2ph: *2pheroG 2phase*
– 2L_scr: *2pheroL scratch*
– 2L_2ph : *2pheroL 2phase*

After Table 14, we will present the results of our experiments question by question and will illustrate these observations with plots of differences of EAFs and relation of preference.

comparison	(50x10-1)	(50x10-2)	(50x30-1)	(50x30-2)
1. Comparison 1phero-2phero approach				
1_scr/2G_scr	2G_scr better	none	1_ scr better in the middle, 2G_scr better in the right	2G_scr better in the right
1_scr/2L_scr	2L_scr better	none	1_scr better in the left, 2L_scr better in the right	none
1_2ph/2G_2ph	2G_2ph better in the left	none	1_2ph better in the middle, 2G_2ph better in the left upper corner	1_2ph better in the middle, 2G_2ph better in the right bottom corner
1_2ph/2L_2ph	2L_2ph better in the left	1_2ph slightly better in the middle, 2L_2ph better in the right	2L_2ph better	1_2ph better in the left upper corner, 2L_2ph in the right
2. Comparison scratch- 2phase approach				
1_scr/1_2ph	1_2ph better	none	1_scr slightly better in the right upper corner, 1_2ph better in the rest	none
2G_scr/2G_2ph	none	none	2G_2ph better	none
2L_scr/2L_2ph	2L_scr better in the middle bottom, 2L_2ph better in the right	2L_2ph better	2L_2ph better	2L_2ph better essentially in the middle and in the right
3. Comparison global-local strategy				
2G_scr/2L_scr	2G better in the right	2G_scr better	2G_scr better in the extreme, 2L_scr better in the middle	2G_scr better in the right, 2L _scr slightly better in the middle
2G_2ph/2L_2ph	none	none	2G_2ph slightly better in the left, 2L_2ph better in the middle and in the right	2L_2ph better in the middle

Table 13 : Summary of the observations for the 4 instances

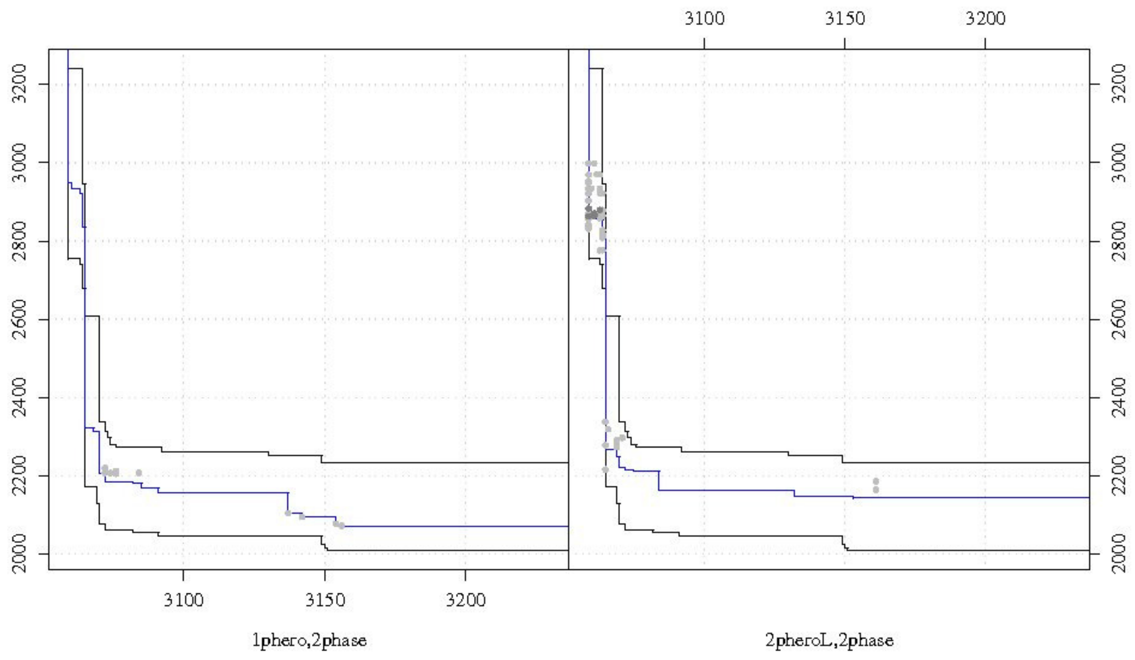


Figure 38 : Differences of EAFs, *1phero 2phase/2pheroL 2phase* (50x10-1)

1. Summary of the observations on *1phero* or 2*phero* approach

– For this question, we have compared:

– *1phero scratch* with *2pheroG scratch*

– *1phero scratch* with 2pheroL scratch

– *1phero 2phase* with *2pheroG 2phase*

– *1phero 2phase* with *2pheroL 2phase.*

For the two instances 50 jobs, 10 machines, we can observe that:

o *2phero* (both *L* and *G) scratch* configuration is never worse than *1phero scratch* and often performs better. More results are presented in appendix 7 (Figures 67, 68).

o *2pheroL 2phase and 2pheroG 2phase* configurations are preferable to *1phero 2phase* when the decision maker looks for solution with best makespan (See Figure 38 illustrates this preference).

For the two instances 50 jobs, 30 machines, the results differ and we can observe that:

o *1phero (scratch* seems to perform better than *2pheroG scratch* in the middle region whereas *2pheroG* seems be better for solutions located in the right bottom corner as

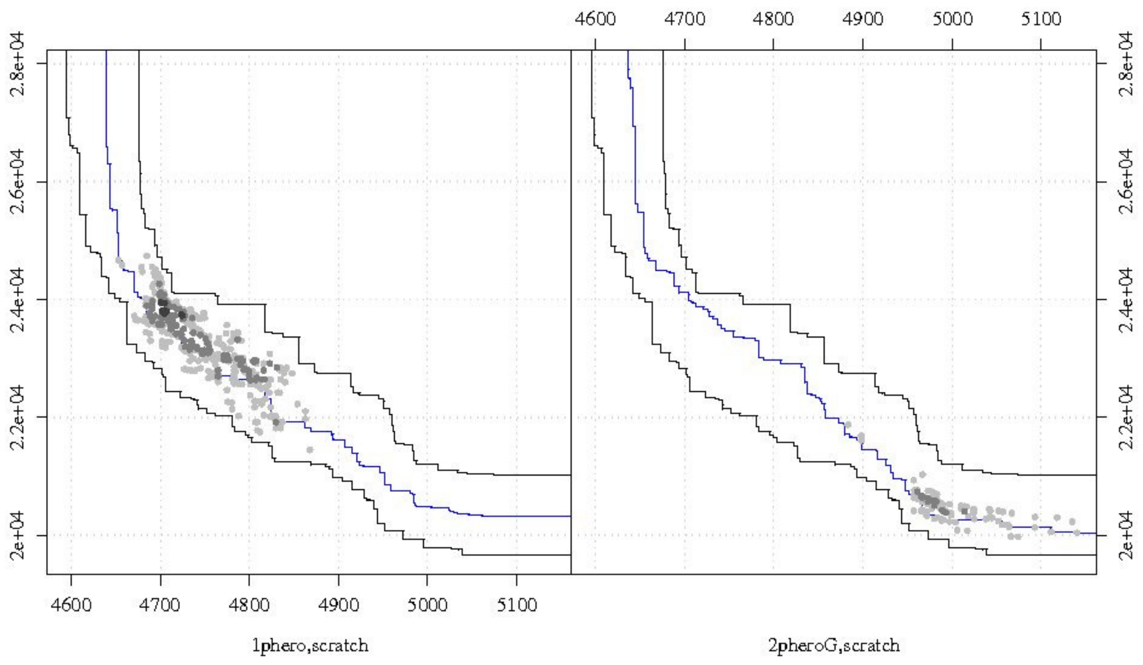


Figure 39: Differences of EAFs, *1phero scratch*/*2pheroG scratch* (50x30-1)

we can see in the figure 39. This figure also gives indications on the poor capability of *2pheroG scratch* to find good solutions in the middle region of the objective space.

- When *1phero 2phase* is compared with *2pheroL 2phase*, we observe that *2pheroL* 2phase seems to perform better than *1phero 2phase* in the middle for a direction changes $\lambda = 1$□ 0 . When the direction changes chosen is $\lambda = 0$□ 1 , the observation is different, *1pero 2phase* seems to be better for most of the regions of the objective space for 50x30-2 (see Figures 75 and 81 in appendix 8). Thus we have an indication that performances of one configuration are function of the direction changes adopted and of the instance tackled.

Conclusion on a comparison *1phero 2phero* approach

- Observations depends on the instances and on the direction changes chosen
- For 50x10 problems, *2pheroG* is never worst than 1phero and is sometimes better
- For 50x30 problems, 1phero better than *2pheroG* in the middle but *2pheroG* better in the right
- For 50x30 problems, *2pheroL* better at least in the right

I_C	(50x10-1)	(50x10-2)	50x30-1	50x30-2
1phero scratch/1phero 2phase	16%	39%	30%	42%
1phero 2phase/1phero scratch	**76%**	49%	60%	48%
2pheroG scratch/2pheroG 2phase	49%	42%	49%	51%
2pheroG 2phase/2pheroG scratch	44%	48%	58%	43%
2pheroL scratch/2pheroL 2phase	36%	12%	17%	29%
2pheroL 2phase/2pheroL scratch	53%	**85%**	**79%**	62%

Table 14 : Coverage measures for a comparison *scratch/2phase*

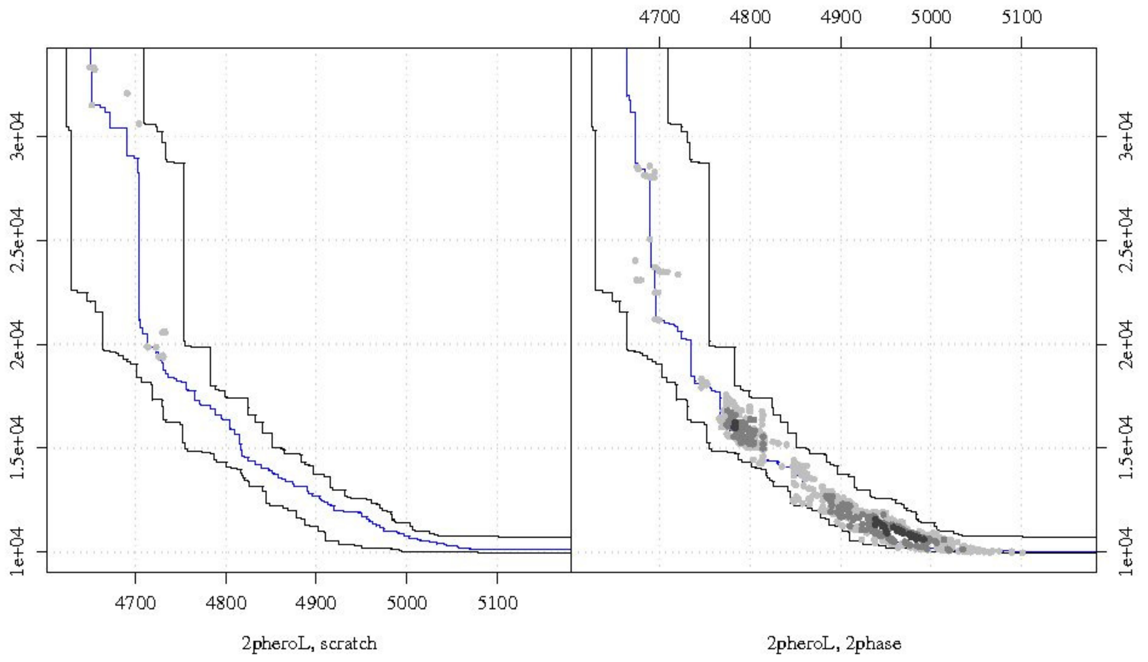


Figure 40: Differences of EAFs, *2pheroG scratch* /*2pheroG 2phase* (50x30-2)

2. Summary of the observations on the *scratch* and the *2phase* approach

For this question, we have compared:

- *1phero scratch* with *1phero 2phase*
- *2pheroG scratch* with *2pheroG 2phase*
- 2pheroL scratch with 2pheroL 2phase

- For the three configurations and for the four instances tested, it appears that the *2phase* approach is never worse than *scratch* approach and sometimes performs better, particularly in the middle and in the right region of the objective space for the configuration *1phero* and the configuration *2pheroL*. Table 13 summarizes the coverage measures for a comparison *scratch-2phase* approach for the four instances tested. The first line of the table represents the percentage of points obtained by the second configuration (*1phero scratch*) which are dominated by at least one point of the second configuration (*1phero 2phase*). The same for the other lines of the table. Percentages in bold show situations where a *2phase* approach is clearly preferable. *2phase* approach is never dominated by a configuration using *scratch* approach and for *2pheroG*, only relation of incomparability appears.

- When it was possible, a visualization of the differences of like Figure 40 and others EAFs presented in appendix 8, have confirmed the impression that a *2phase* approach is preferable even if it does not dominate *scratch* approach. In Figure 40, we observe that 2pheroL 2phase performs better than *2pheroL scratch* in the right region of the objective space and is not worse in the other regions.
- For *2pheroL* configuration, the *2phase* approach seems particularly important to find good quality solutions. This configuration if compared with *2pheroG* is more oriented towards the exploration of the solution space than on the exploitation of the information to improve the quality of the solution. Thus beginning each phase with a solution of good quality can be very useful.
- For the *2pheroG* approach, choosing between *scratch* or *2phase* approach does not clearly affect the performance. The structure of this algorithm could be one of the reasons why the few differences appear between all the configurations.
- *2pheroG* focus more on the improvements of the quality of the solutions than the two other configurations. Thus starting the process with a good solution seems to be less important.

Conclusion on a comparison scratch-2phase approach

- *2phase* approach is never worse than *scratch* approach and sometimes performs better, particularly in the middle and in the right region of the objective space for the configuration *1phero* and the configuration *2pheroL*

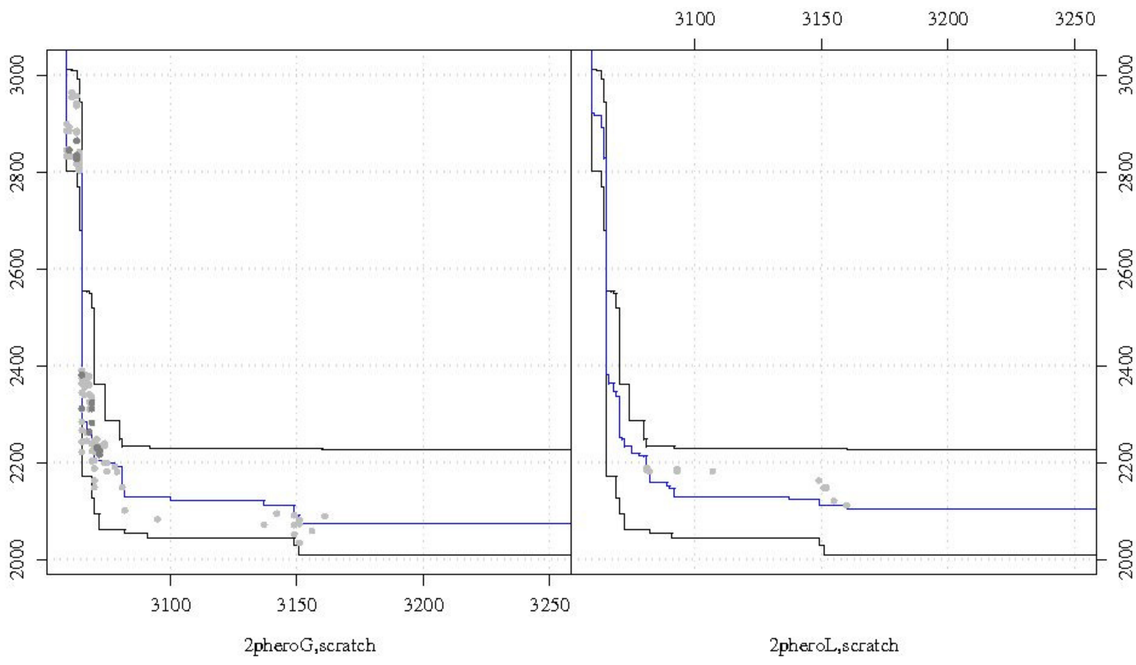


Figure 41: Differences of EAFs, *2pheroG* scratch/*2pheroL* scratch (50x10-1)

Summary of the observations on the *global* and *local* strategy

For the two instances 50 jobs, 10 machines, we can observe that:

- *2pheroG scratch* seems to be generally more performing than *2pheroL scratch* at least if the decision maker prefers solutions with a small makespan. In the Figure 41 it is possible to observe two groups of points in the left side of the left plot. It is difficult to know if this difference is due to the algorithm, to the type of direction changes used for the tests or both factors.
- There is no indications that applying *2pheroG 2phase* or *2pheroL 2phase* to the problem gives better solutions for one region or another in the objective space.

For the two instances 50 jobs. 30 machines, the observations are different, we can observe that:

- *2pheroLscratch* seems to be more capable of finding good solutions in the middle region of the objective space while *2pheroGscratch* seems to be better at the extreme of the Pareto front. This expected result can be observed in Figure 42.

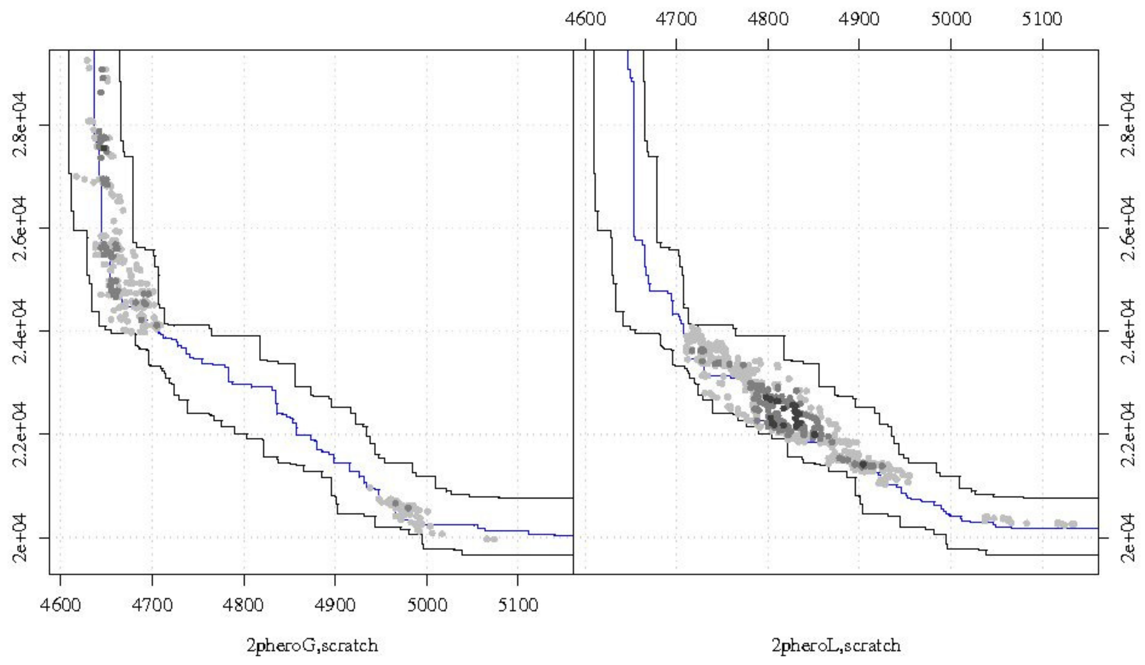


Figure 42: Differences of EAFs, *2pheroG scratch/2pheroL scratch* (50x30-1)

- With *2phase* approach, *2pheroL* seems to be better for most regions, except for a small region in the left upper corner, where the makespan is the smallest and when the type of direction changes chosen begins with $\lambda = 1$. But when the direction changes chosen begins with $\lambda = 0$ the result of the comparison can be opposite. This is with Tables 56 and 58 in appendix 9.

Conclusion on a comparison global-local strategy

- Observations depends on the instances and on the direction changes chosen
- For 50x10 problems, *2pheroG* is never worst than *2pheroL* and is sometimes better when a *scratch* approach has been chosen

For 50x30 problems, 2pheroL is better in the middle

6

Conclusion

ACO metaheuristic, a method of optimization inspired by foraging behavior of real ants, has already shown very good performance for many combinatorial optimization problems and for some real applications. In this master thesis, contrarily to most of works on flowshop problems which are single objective approach, we have studied a biobjective permutation flowshop scheduling problem where the makespan and the total tardiness are the objectives to optimize.

In this work we have first briefly reviewed the different methods of optimization which have been applied to the single objective flowshop problem with makespan and total tardiness objective tackled separately. We have also reviewed different techniques applied to multiobjective optimization problems like genetic algorithms, tabu search, ACO or simulated annealing. We have focused essentially on ACO algorithms. ACO algorithms dedicated to multiobjective problems can be classified in three categories, one single colony and multiple pheromone matrices, multiple colonies and one pheromone matrix and multiple colonies and multiple pheromone matrices. These ACO algorithms have been applied to multiple problems such as the water distribution design, a vehicle routing problem with time window, bicriteria travel salesman problem, single machine total tardiness problem with changeover costs, etc.

We propose two different ACO algorithms based on MMAS with the integration of the summation rule proposed by Middendorf. These two algorithms use multiple colonies and works respectively with one and two pheromone matrices to solve the biobjective flowshop problem. The idea of these algorithms is to exploit the good performances of ACO algorithms for single objective problems. In these algorithms, each colony is forced to search for solutions in different directions of the search space by changing the importance of each objective all along the procedure.

These two algorithms and their different variants have been tested on four instances and compared through the analysis of more than one hundred pairwise comparisons.

Before presenting the results obtained, it is important to notice that the results of our tests can only provide indications and suggestions. More experiments on more instances, during a longer computation time can be necessary for more consistent conclusions. Our analysis provides indications on an eventual strategy to adopt to tackle this biobjective flowshop problem following the preferences of the decision maker and the kind of instance. The results also provide indications and suggestions for further future researches.

Actually our comparisons on the performance of the two proposed ACO algorithms and their variants have faced two main problems. The performance of the different configurations depends strongly on the kind of instance and on the direction changes chosen. Results obtained for the instances with 50 jobs and 10 machines clearly differ from those obtained for the instances with 50 jobs and 30 machines, and observations resulted from a comparison between two configurations may sometimes be different or opposite following the direction changes used during the procedure. Thus general and global conclusions on the performance of the different configurations are very difficult. Nevertheless some suggestions are possible and we will now expose them:

- First, we have observed that a certain number of aggregation weights is necessary to obtain a good approximation of the Pareto front but it appears that enough computation time must also be allocated to the search of solutions. Too few numbers of weights and points in the middle will not be found, too large number of weights and the quality of the solution will be poor because the algorithm will not have enough time to construct good solutions.
- Secondly, the results of our experiments suggest that the *2phase* approach has a positive effect for a majority of the configurations and the instances tested. It seems that the solution obtained for one aggregation weight is already an acceptable solution for the next aggregation weight if the change is minor, thus a *2phase* approach helps to find a set of non dominated solutions of higher quality.
- Thirdly the results of the comparisons suggest that some configurations are more capable to find non dominated solutions in certain regions of the objective space. Thus following eventual preferences of the decision maker on the objective, one configuration may be preferred.

For future works, many experiments can improve the understanding of the problem and of the behaviour of the different algorithms and their variants:

- More experiments to determine more precisely the influence of the different types of directions changes on the performance of each algorithm would be very interesting. This kind of study would help to choose the best couple (algorithm, direction changes) for solving a multiobjective problem
- A comparison of the performance of the algorithm we proposed with other multiple objective optimizers using genetic algorithm, tabu search or simulated annealing which have already showed good performance for other multiobjective problem, will also provides good information on the relevance of using ACO algorithms to solve a biobjective permutation flowshop scheduling problem.
- Experiments on instances with different coefficient of correlation between the two objectives in addition of tests on instances with different number of machines and jobs will provide information on the influence of the kind of instance on the performance of the different algorithms.
- Finally it can be noticed that *2phase* approach is not specific to ACO algorithms. A study on the advantages of this *2phase* approach used with other kind of metaheuristics and algorithms to tackle multiobjective optimization problems could be very interesting for an eventual generalisation.

Bibliography

1 B. Adenso-Díaz, *An SA/TS Mixture Algorithm for The Scheduling Tardiness Problem*, European Journal of Operational Research, 88, 516-524, 1996.

2 J. Admas, E. Balas and D. Zawack, *The Shifting Bottleneck Procedure in Job Shop Scheduling,* Management Science,34(3):391-401, 1988.

3 A. Allahverdi, *The two- and m-machine flowshop scheduling problems with bicriteria of makespan and mean flowtime*, European Journal of Operational Research, 147(2):373–396, 2003.

4 A. Al-Yamani., S. Sait and H. Youssef, *Parallelizing Tabu Search on a Cluster of Heterogeneous Workstations*, Journal of Heuristics on Parallel Metaheuristics, 8(3), pp. 277-304, 2002.

5 B. Barán and S. Duarte, Multiobjective *Network Design Optimization using Parallel Evolutionary Algorithms*, Centro Nacional de Computación, Universidad Nacional de Asunción. San Lorenzo, Paraguay. Agosto 2002.

6 B. Barán, M. Schaerer, *A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows*, Proc. Twenty first IASTED International Conference on Applied Informatics, Innsbruck, Austria, February 10-13, pp. 97-102, 2003.

7 R.P. Beausoleil, *Multiple criteria scatter search*, Proceedings of the 4th Metaheuristics International Congress, Porto, Portugal, pp. 539-543, 2001.

8 C. Blum and M. Dorigo, *The hyper-cube framework for ant colony optimization*, IEEE Transactions on Systems, Man, and Cybernetics – Part B, vol. 34, no. 2, pp. 1161–1172, 2004.

9 C. Blum, A. Roli, and M. Dorigo, *HC–ACO: The hyper-cube framework for Ant Colony Optimization,* Proceedings of MIC'2001 –Metaheuristics International Conference, vol. 2, Porto, Portugal,, pp. 399–403, 2001.

10 C. Blum, *Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling*, Computers & Operations Research, vol. 32, no. 6, pp. 1565–1591, 2005.

11 K. D. Boese, A. B. Kahng, and S. Muddu, *A new adaptive multi-start technique for combinatorial global optimization*, Operations Research Letters, 16:101–113, 1994.

102M. Bolondi and M. Bondanza, *Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore*, Master's thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1993.

13 M. Bonney and S. Gundry, *Solutions to the constrained flowshop sequencing problem*, Operational Research Quarterly 27 (4), 869–883, 1976.

14 Basseur, F. Seynhaeve and E.G. Talbi, Design *of multiobjective evolutionary algorithms: Application to the flow-shop scheduling problem*, CEC'2002 Congress on Evolutionary Computation, pp.1151-1156, Hawaii, Mai 2002

15 M. Basseur, F. Seynhaeve and E.G. Talbi, *Adaptive mechanisms for multi-objective evolutionary algorithms* , CESA'2003 Computational Engineering in Systems Applications IEEE Press, Lille, France, Juil. 2003.

16 M. Basseur, F. Seynhaeve and E-G. Talbi, Path *relinking in Pareto multiobjective optimization algorithms*, Int. Conf. On Evolutionary Multicriterion Optimization, Lectures Notes in Computer Science Lectures Notes in Computer Science LNCS No.3410, Edited by C .Coello et al., pp.120-130, Guanajuato, Mexico, Mar 2005.

17 B. Bullnheimer, R. F. Hartl, and C. Strauss*, A new rank based version of the Ant System — a computational study,* Institute of Management Science, University of Viena, Tech. Rep., 1997.

18 B. Bullnheimer, G. Kotsis, and C. Strauss, *Parallelization strategies for the Ant System*, R. De Leone, A. Murli, P. Pardalos, and G. Toraldo, editors, High performance Algorithms and Software in Nonlinear Optimization, volume 24 of Applied Optimization, pages 87–100. Kluwer Academic Publishers, Dordrecht, NL, 1998.

19 B. Bullnheimer, R.F. Hartl, and C. Strauss, *Applying the ant system to the vehicle routing problem*, Voss S., Martello S., Osman I.H., and Roucairol C. (eds.) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Boston, 1999.

20 E.K. Burke, P. Cowling, J.D. Landa Silva, S. Petrovic *Combining Hybrid Metaheuristics and Populations for the Multiobjective Optimisation of Space Allocation Problems*, Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO 2001), San Francisco USA, Morgan Kaufmann, pp. 1252-1259, 2001.

21 H.G. Campbell, R. A. Dudek, M. L. Smith, *A heuristic algorithm for the n job, m machine sequencing problem*, Management Science 16 (10), B630–B637, 1970.

22 P. Cardoso, M. Jesús, A. Márquez, *MONACO - Multi-Objective Network Optimisation Based on an ACO*, Proc. X Encuentros de Geometría Computacional, Seville, Spain, June 16-17, 2003.

23 O. Cepek, M. Okada and M. Vlach., *Nonpreemptive flowshop scheduling with machine dominance*, European Journal of Operational Research, 139(2):245–261, 2002.

24 V. Cerný, *Thermodynamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm*, J. Opt. Theory Appl., 45, 1, 41-51, 1985.

25 A. Chattopadhyay and C. E. Seeley, *A simulated annealing technique for multiobjective optimization of intelligent structures*, Smart Materials and Structures, Volume 3, Number 2,, pp. 98-106(9), 1994.

26 C.L. Chen, V.S. Vempati and N. Aljaber*, An application of genetic algorithms for flow shop problems*, European Journal of Operational Research 80, 389–396, 1995.

27 A. Coello, S.M. Reyes, *A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm*, MICAI : 688-697, 2004.

28 A. Colorni ,M. Dorigo, V. Maniezzo and M. Trubian, *Ant system for job-shop scheduling*, Belgian Journal of Operations Research, Statistics and Computer Science, 34(1): 39-54, 1994

29 O. Cordón, I. F. de Viana, F. Herrera, and L. Moreno, *A new ACO model integrating evolutionary computation concepts: The best-worst Ant System*, Abstract proceedings of ANTS 2000 – From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms, M. Dorigo et al., Eds. IRIDIA, Université Libre de Bruxelles, Belgium,, pp. 22–29, 2000;

30 D. Costa and A. Hertz, *Ants can colour graphs*, Journal of the Operational Research Society, vol. 48, pp. 295–305, 1997.

31 T. G. Crainic and M. Toulouse, Parallel *Strategies for Metaheuristics, in State-of-the-Art Handbook* in Metaheuristics, F. Glover, G. Kochenberger (Eds.), Kluwer Academic Publishers, 2002

32 V.-D.Cung, S.L. Martins, C.C. Ribeiro, and C. Roucairol, *Strategies for the parallel implementation of metaheuristics*, Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, Dordrecht. pp. 263-308, 2003

33 K. Dahl, K. Jornsten and A. Lokketangen, *A tabu search approach to the channel minimization problem*, G. Liu, K-H. Hua, J. Ma, J. Xu, F. Gu and C. He, editors, Optimization- Techniques and Applications, ICOTA '95, Volume 1, pages 369-377, Chengdu, China, 1995. World Scientific.

34 D.G. Dannenbring, *An evaluation of flow shop sequencing heuristics*, Management Science 23 (11), 1174–1182, 1977.

35 C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, London: John Murray 1st edition. Retrieved on 2006-12-31, 1859.

36 M.L. den Besten, T. Stützle and M. Dorigo, *Scheduling single machines by ants*, Technical report IRIDIA/99-16, Institutde Recherches Interdisciplinaires et de Developpements en Intelligence Artificelle, Université´ Libre de Bruxelles, Brussels, Belgium, 1999.

37 M. L. den Besten, T. Stützle, and M. Dorigo, *Ant colony optimization for the total weighted tardiness problem*, Proceedings of PPSN-VI, ser. LNCS, M. Schoenauer et al., Eds., vol. 1917. Springer Verlag, pp. 611–620, 2000.

38 P. Delisle, M. Krajecki, M. Gravel and C. Gagné, *Parallel implementation of an ant colony optimization metaheuristic with OpenMP, International Conference on Parallel Architectures and Compilation Techniques*, Proceedings of the 3rd European Workshop on OpenMP (EWOMP'01), 8-12 septembre 2001, Barcelone, Espagne, 2001.

39 F. de Toro, J. Ortega , E. Ros , S. Mota , B. Paechter and J. M. Martín, *PSFGA: parallel processing and evolutionary computation for multiobjective optimisation*, Parallel Computing, v.30 n.5-6, p.721-739, May 2004

40-J. L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels, *The self organizing exploratory pattern of the Argentine ant*, Journal of Insect Behaviour, vol. 3, p. 159, 1990.

41 G. Di Caro and M. Dorigo, *AntNet*: *A mobile agents approach to adaptive routing*, Technical Report 97-12, IRIDIA, Université Libre de Bruxelles, 1997.

42 G. Di Caro and M. Dorigo, *Two ant colony algorithms for best-effort routing in datagram networks*, Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98), pages 541-546 IASTED/ACTA Press, 1998.

43 G. Di Caro and M. Dorigo, *AntNet: Distributed stigmergetic control for communications Networks*, Journal of Artificial Intelligence Research (JAIR), 9:317–365, December 1998.

44 K. Doerner, J. Gutjahr, R. Hartl, C. Strauss, and C. Stummer, *Investitionsentscheidungenbei mehrfachen Zielsetzungen und künstliche Ameisen*, In Chamoni and et al., editors, Operations Research Proceedings, p. 355-362, Berlin, Heidelberg.2001. Springer.

45 K. Doerner, J. Gutjahr, R. Hartl, C. Strauss, and C. Stummer, *Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection,* European Journal of Operational Research. 171(3), pp. 830-841, 2006

46 K. Doerner, R. Hartl and M. Reimann. *Are COMPETants more competent for problem solving? - the case of a multiple objective transportation problem*, In L. S. et al., editor, Proceedings of the GECCO'01, p.802, Berlin, Heidelberg, 2001. Morgan Kaufmann.

47 K. Doerner, R. Hartl, and M. Reimann. *Cooperative ant colonies for optimizing resource allocation in transportation,* In W. Boers and et al., editors, Proceedings of the EvoWorkshops 2001, p. 70-79, Berlin, Heidelberg, 2001. Springer.

48 M. Dorigo, *Optimization, learning and natural algorithms (in Italian), Ph.D. dissertation*, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

49 M. Dorigo, M. Birattari and T. Stützle, *Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique*, IEEE Computational Intelligence Magazine, volume 1, numéro 4, pages 28-39, 2006.

50 M. Dorigo, G. Di Caro, *The ant colony optimization meta-heuristic*, D. Come, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, pp. 11–32, 1999.

51 M. Dorigo, G. Di Caro and L.M. Gambardella, Ant *Algorithms for Discrete Optimization*, Artificial Life, Vol.5, No.3, pp. 137-172, 1999

52 M. Dorigo and L. M. Gambardella, *Ant colonies for the travelling salesman problem*, Bio Systems, vol. 43, no. 2, pp. 73–81, 1997.

53 M. Dorigo and L. M. Gambardella, *Ant Colony System: A cooperative learning approach to the travelling salesman problem*, IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53–66, 1997

54 M. Dorigo, V. Maniezzo, and A. Colorni, P*ositive feedback as a search strategy,* Dipartimento di Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016, 1991.

55 M. Dorigo, V. Maniezzo, and A. Colorni, *Ant System: Optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics – Part B, vol. 26, no. 1, pp. 29–41, 1996.

56 R.A. Dudek., O.F. Teuton Jr., *Development of m-stage decision rule for scheduling n jobs through m machines*, Operations Research 12 (3), 471–497, 1964.

57 C.J. Eyckelhof and M. Snoek, *Ant systems for a dynamic TSP: Ants caught in a traffic jam*, Proc. ANTS 2002, ser. LNCS, M. Dorigo et al., Eds., Springer Verlag, vol. 2463, pp 88–99, 2002.

58 C. M. Fonseca, *Multiobjective Genetic Algorithms with Application to Control Engineering Problems*. PhD thesis, University of Sheffield, 1995.

59 M.Fonseca and P.J.Fleming , *Genetic algorithms for multiobjective optimizations, Formulation, discussion and generalization*, Proc.of 5th Int.Conf.on Genetic Algorithms(ICGA'93), p.416-423, San Mateo, USA, 1993.

. 60 C. M. Fonseca and P. J. Fleming, *On the performance assessment and comparison of stochastic multiobjective optimizers,* H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, Parallel Problem Solving from Nature—PPSN IV, Lecture Notes in Computer Science, pages 584-593, Berlin, Germany, 1996. Springer-Verlag.

61 Gagné, M. Gravel and W.L. Price, *Scheduling a single machine where setup times are sequence dependent using an ant-colony Heuristic*, in Abstract Proceedings of the International Workshop on Ant Algorithms, M. Dorigo, L.M. Gambardella, M. Middendorf and T. Stützle, editors, 2000, pp. 157–160, 2000.

62 C. Gagné, M. Gravel and W.L. Price, *Scheduling continous casting of aluminium using a multiple objective ant colony optimization heuristic*, European Journal of Operational Research, 143 p. 218-229, 2002

63 L. M. Gambardella and M. Dorigo, *Ant-Q: A reinforcement learning approach to the travelling salesman problem*, Proceedings of the Twelfth International Conference on Machine Learning (ML-95), A. Prieditis and S. Russell, Eds. Morgan Kaufmann Publishers, pp. 252–260, 1995.

64 L. M. Gambardella and M. Dorigo, *Solving symmetric and asymmetric TSPs by ant colonies*, Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC’96), T. Baeck et al., Eds. IEEE Press, Piscataway, NJ,, pp. 622–627, 1996.

65 L. M. Gambardella and M. Dorigo, *HAS-SOP: An hybrid ant system for the sequential ordering problem*, Technical Report 11-97, IDSIA, Lugano, CH, 1997.

646L. M. Gambardella and M. Dorigo, *Ant Colony System hybridized with a new local search for the sequential ordering problem*, INFORMS Journal on Computing, vol. 12, no. 3, pp. 237–255, 2000.

67 L. M. Gambardella, E. D. Taillard, and M. Dorigo, *Ant colonies for the QAP.*, Technical Report 4-97, IDSIA, Lugano, Switzerland, 1997.

68 L. M. Gambardella, E. D. Taillard, and G. Agazzi, *MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows*, New Ideas in optimization, D. Corne et al., Eds. McGraw Hill, London, UK, , pp. 63–76, 1999.

69 Gandibleux and M. Ehrgott, *1984-2004 – 20 Years of Multiobjective Metaheuristics. But what about the Solution of Combinatorial Problems with Multiple Objectives?,* EMO'05 — March 9-11, 2005. Guanajuato, Mexico

70 X. Gandibleux, , N. Mezdaoui, A. Freville, , R. Caballero, F. Ruiz, and R. Steuer, *A tabu search procedure to solve multiobjective combinatorial optimization problems*, Advances in Multiple Objective and Goal Programming, Lecture Notes in Economics and Mathematical Systems, v455, Springer, Berlin, Germany, pp. 291-300, 1997.

71 M. Garey and D. Johnson, *Computers and Intractability*, WH Freeman, San Francisco, 1979.

72 L.F. Gelders and N. Sambandam, *Four simple heuristics for scheduling a flowshop*, International Journal of Production Research, 16, 221-231, 1978.

73 D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Kluwer Academic Publishers, Boston, MA,1989

74 R. Graham, E. Lawler, J. Lenstra, and A. R. Kan, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, *Annals of Discrete Mathematics*, vol. 5, 1979, pp. 287–326.

75 M. Guntsch, M. Middendorf, Solving *Multi-criteria Optimization Problems with Population-Based ACO*, C.M. Fonseca and P.J. Fleming and E. Zitzler and K. Deb and L. Thiele, *Evolutionary Multi-Criterion Optimization, Second International Conference (EMO'03)*, no. 2632 in LNCS, p. 464-478. Springer, Berlin, Heidelberg, 2003

76 M. Guntsch and M. Middendorf. *A population based approach for ACO*, S. Cagnoni and et al., editors, Applications of Evolutionary Computing -EvoWorkshops 2002:

EvoCOP, EvoIASP, EvoSTIM/EvoPLAN, number 2279 in LNCS, p. 72-81. Springer, 2002.

77 J.N. Gupta, *A functional heuristic algorithm for the flowshop scheduling problem*, Operational Research Quarterly 22 (1), 39–47, 1971.

78 M. P. Hansen, Tabu *search for multiobjective optimization: MOTS*, 13th Internat. Conf. Multiple Criteria Decision Making, 1997.

79 A. Hertz, B. Jaumard, C. Ribeiro and F. W. Formosinho, *A multicriteria tabu search approach to cell formation problems in group technology with multiple objectives*, Recherche Operationelle/Oper. Res., v28, pp. 303-328, 1994.

80 J.C. Ho and Y.L. Chang, *A new heuristic for the n-job, m machine flow-shop problem*, European Journal of Operational Research 52, 194–202, 1991.

81 J. H. Holland, Adaptation *in Natural and Artificial Systems*, University of Michigan, Press, Ann Arbor, 1975

82 T.S. Hundal and J. Rajgopal, *An extension of Palmer's heuristic for the flow shop scheduling problem*, International Journal of Production Research 26 (6), 1119–1124, 1988.

83 L. Hwang and A. S. M. Masud, *Multiple objective decision making-methods and applications*, Lecture Notes in Economics and Mathematical systems, volume 164. Springer-Verlag , Berlin, 1979.

84 S. Iredi, D. Merkle, and M. Middendorf, *Bi-criterion optimization with multi colony ant algorithms*, E. Zitzler and et al., editors, Evolutionary Multi-Criterion Optimization, First International Conference (EMO'01), number 1993 in LNCS, p. 359-372, Berlin, Heidelberg; 2001. Springer.

85 H. Ishibuchi, S. Misaki, H. Tanaka, *Modified simulated annealing algorithms for the flow shop sequencing problem*, European Journal of Operational Research 81, 388–398, 1995.

86 A. Jaszkiewicz, *Genetic local search for multi-objective combinatorial optimization*, European J. of Operational Research, vol. 137, pp. 50-71, 2002

87 S. Johnson, *Optimal two and three stage production schedules with setup times included*, Naval Research Logistics Quaterly, 1(1):61–68, 1954.

88 N. Jozefowiez , *Modélisation et résolution approchées de problèmes de tournées multicritères*, PhD thesis, University of Lille, Lille, France, 2004

89 N. Jozefowiez , F. Semet,E-G. Talbi, *Parallel and hybrid models for multi-objective optimization: Application to the VehicleRouting Problem*, Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN VII), Number 2439 in Lecture Notes in Computer Science, p. 271-280, Granada, Spain, September 2002 Springer-Verlag,

90 Y.D. Kim, *Heuristics for Flowshop Scheduling Problems Minimizing Mean Tardiness*, *Journal of the Operational Research Society*, Vol. 44, No. 1, pp. 19-28, 1993.

91 Y.D. Kim, G. Lim and M.W. Park, *Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process,* European Journal of Operational Research, Volume 91, Number 1 , pp. 124-143(20), 1996.

92 Kirkpatrick, S., C.D. Gelatt Jr. and M. P. Vecchi, *Optimization by Simulated nnealing*,Science, 220, 4598, 671-680, 1983.

93 J. Knowles, L. Thiele, and E. Zitzler, *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*, TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, February 2006.

94 N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon and F. Soumis*, K-Path Cuts for the Vehicle Routing Problem with Time Windows,* Technical Report IMM-REP-1997-12, Technical University of Denmark, 1997.

95 C. Koulamas, *A new constructive heuristic for the flowshop scheduling problem*, European Journal of Operational Research Society 105, 66–71, 1988.

96 F. Krüger, D. Merkle, and M. Middendorf*, Studies on a parallel ant system for the BSP model*, Unpublished manuscript.

97 G. B. Lamont and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 2002.

98 D. A. Linkens and H. Okola, A *distributed genetic algorithm for multivariable fuzzy control,* Proc. IEE Colloquium on Genetic Algorithms for Control and Systems Engineering, volume 130, pages 9/1--9/4. London **, May 1993.

99 M. López-Ibáñez, L. Paquete, and T. Stützle, *Hybrid population-based algorithms for the bi-objective quadratic assignment problem,* Journal of Mathematical Modelling and Algorithms, 5(1):111-137, April 2006.

100 V. Maniezzo, *Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem*, INFORMS Journal on Computing, vol. 11, no. 4, pp. 358–369, 1999

101 V. Maniezzo, A. Colorni, and M. Dorigo, *The ant system applied to the quadratic assignment problem*, Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994.

102 C. E. Mariano and E. Morales, *MOAQ an ant-Q algorithm for multiple objective optimization problems,* W. Banzhaf and et al., editors, Proceedings of the Genetic and Evolutionary Computation Conference, volume 1, p. 894-901, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

103 R. Michel and M. Middendorf, *An island model based ant system with lookahead for the shortest supersequence problem*, A. E. Eiben, T. Back, M. Schoenauer and H.-P. Schwefel, editors, Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, pages 692–701. Springer-Verlag, 1998.

104 D. Merkle and M. Middendorf, *An ant algorithm with a new pheromone evaluation rule for total tardiness problems*, Proceedings of the EvoWorkshops 2000. In: Lecture Notes in Computer Science, vol. 1803. Springer-Verlag,Berlin,pp. 287–296, 2000

105 D. Merkle and M. Middendorf, *Ant colony optimization with global pheromone evaluation for scheduling a single machine*, Applied Intelligence, vol. 18, no. 1, pp. 105–111, 2003.

106 D. Merkle, M. Middendorf, and H. Schmeck, *Ant colony optimization for resource-constrained project scheduling*, IEEE Transactions on Evolutionary Computation, vol. 6, no. 4, pp. 333–346, 2002.

107 R. Michel and M. Middendorf, *An island model based ant system with lookahead for the shortest supersequence problem*, A. E. Eiben, T. Back, M. Schoenauer and H.-P. Schwefel, editors, Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, pages 692–701. Springer-Verlag, 1998.

108 R. Michel and M. Middendorf, *An ACO algorithm for the shortest supersequence Problem,* D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 51–61. McGraw Hill, London, UK, 1999.

109 M. Middendorf, F. Reischle, and H. Schmeck, *Information exchange in multi colony ant algorithms*, J. Rolim, editor, Parallel and Distributed Computing, Proceedings of the 15 IPDPS 2000 Workshops, Third Workshop on Biologically Inspired Solutions to Parallel Processing Problems (BioSP3), volume 1800 of Lecture Notes in Computer Science, pages 645–652. Springer Verlag, Berlin, Germany, 2000.

110 J.a.V. Moccellin and M.O. dos Santos, *An adaptive hybrid meta-heuristic for permutation flowshop scheduling*, Control and Cybernetics 29 (3), 761–771, 2000.

111 R. Montemanni, L.M. Gambardella, A.E. Rizzoli, and A.V. Donati, *Ant colony system for a dynamic vehicle routing problem*, Journal of Combinatorial Optimization, vol. 10, pp. 327–343, 2005.

112 T. Morton, DW. Pentico, *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management*, Wiley, 1993.

113 T. Murata, , H. Ishibuchi and H. Tanaka, *Genetic algorithms for flowshop scheduling problems*, Computers and Industrial Engineering 30 (4), 1061–1071, 1996.

114 M. Nawaz, E.E. Enscore Jr.and I. Ham, *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, OMEGA, The International Journal of Management Science 11 (1), 91–95, 1983.

115 E.Nowicki and C. Smutniki, *A fast tabu search algorithm for the permutation flowshop problem*, European Journal of Operational Research, 91, 160-175, 1996.

116 F. Ogbu, D. Smith, *The application of the simulated annealing algorithms to the solution of the n=m=Cmax flowshop problem*, Computers and Operations Research 17 (3), 243–253, 1990.

117 G.C. Onwubolu and M. Mutingi, *Genetic algorithm for minimizing tardiness in flow-shop scheduling*, Production Planning and Control, Volume 10, Number 5, 1 July 1999 , pp. 462-471(10), 2999.

118 Osman and C. Potts, *Simulated annealing for permutation flow-shop scheduling, OMEGA, The International Journal of Management Science 17 (6), 551–*557, 1989.

119 E. S. Page, An *approach to the scheduling of jobs on machines*, Journal of the Royal Statistical Society, B Series23 (2), 484–492, 1961.

20 PS. Ow, *Focused Scheduling in Proportionate Flowshops, Management Science* 31(7):852-869, 1985.

121 D. Palmer, *Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum*, Operational Research Quarterly 16 (1), 101–107, 1965.

122 Paquete and T. Stützle, *A two-phase local search for the biobjective traveling salesman problem,* C. M. Fonseca, P. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, Evolutionary Multi-criterion Optimization (EMO 2003), volume 2632 of Lecture Notes in Computer Science, pages 479-493. Springer Verlag, 2003. ((c) Springer Verlag).

123 V. Pareto, *Cours d'Economie Politique*. Rouge, Lausanne, Switzerland, 1896.

124 S. Parthasarathy and C. Rajendran, *A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs-a case study*, Production Planning & Control, Volume 8, Issue , pages 475 – 483, 1997.

25 S. Parthasarathy and C. Rajendran, *An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs*, International Journal of Production Economics, Volume 49, Number 3, pp. 255-263(9), 1997.

126 C.N. Potts, D.B. Shmoys and D.P. Williamson, *Permutation vs. non-permutation flow shop schedules*, Operations Research Letters 10, 281–284, 1991.

127 B.J. Pretzel, W. Eheart, S. Rajasthan, *Using genetic algorithm to solve a multiple objective groundwater pollution containment problem.* Water Resources Research, 30, pp. 1589-1603, 1994.

128 D. Quagliarella and A. Vicini, *Genetic algorithms and evolution strategy sin engineering design*, Section Coupling genetic algorithms and gradient based optimization techniques, p.289-309, john Wiley and Sons, Sussex, England, 1997.

129 C. Rajendran & H. Ziegler, *Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs*, European Journal of Operational Research, 155, 426-438, 2004

130 C.R. Reeves, *Improving the efficiency of tabu search for machine scheduling problems*. Journal of the Society 44 (4), 375–382, 1993.

131 C.R. Reeves, C.R., A *genetic algorithm for flowshop sequencing*, Computers and Operations Research 22 (1), 5–13, 1995.

132 M. Reimann, K. Doerner, and R. F. Hartl, *D-ants: Savings based ants divide and conquer the vehicle routing problems*, Computers & Operations Research, vol. 31, no. 4, pp. 563–591, 2004.

133 J. L. Rogers, *A Parallel Approach to Optimum. Actuator Selection With A Genetic Algorithm*, AIAA Guidance, Navigation and Control Conf., 2000.

134 J. Rowe, K. Vinsen and N. marvin, *Parallel GAs for Multi-objective Functions*, jarmo T. Alander, editor, Proceedings of the Second Nordic Workshop on Genetic Algorithms and Their Applications (2NWGA), pages 61-70, Vaasa, Finland, August 1996

135 R. Ruiz and C. Maroto, *A comprehensive review and evaluation of permutation flowshop heuristics*, European Journal of Operational Research, 165, 479-494, 2005.

136 S. Sarin and M. Lefoka, Scheduling *heuristic for the n-job m-machine flow shop*. OMEGA, The International Journal of Management Science 21 (2), 229–234, 1993.

137 J. D. Schaffer, *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, Proceedings of the 1st International Conference on Genetic Algorithms, p.93-100, July 01, 1985.

138 R. Schoonderwoerd, O. Holland, and J. Bruten, *Ant-like agents for load balancing in telecommunications networks*, Proceedings of the First International Conference on Autonomous Agents, pages 209–216. ACM Press, 1997.

139 R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, *Ant-based load balancing in telecommunications networks*, Adaptive Behavior, 5(2):169–207, 1996.

140 P. Serafini, *Simulated annealing for multiple objective optimization problems*? Proceedings of the Tenth International Conference on Multiple Criteria Decision Making, Taipei 19-24.07.92, vol. 1, 87-96. 1992

141 K. Socha, M. Sampels, and M. Manfrin, *Ant algorithms for the university course timetabling problem with regard to the state-of the- art*, Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2003, ser. LNCS, G. R. Raid et al., Ets., vol. 2611. Springer Verlag, pp. 334–345, 2003.

142 T. Stützle, Applying *iterated local search to the permutation flow shop problem*, Technical Report, AIDA-98-04, FG Intellektik, TU Darmstadt, 1998.

143 T. Stützle, *Parallelization strategies for ant colony optimization*, Agoston E. Eiben, Thomas Bäck, M. Schoenauer, and H-P. Schwefel, editors, Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, volume 1498 of Lecture Notes in Computer Science, pages 722–731. Springer Verlag, Berlin, Germany, 1998.

144 T. Stützle and M. Dorigo, *Aco algorithms for the quadratic assignment problem*, New Ideas in Optimization, McGraw-Hill, London, pp. 3—50, 1999.

145 T. Stützle and H. Hoos, *Improvements on the ant system: Introducing MAX–MIN ant system,* Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, pages 245–249, Springer Verlag, Wien, 1997.

146 T. Stützle and H. Hoos, *The MAX–MIN Ant System and local search for the travelling salesman problem*, Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), T. Bäck et al., Eds. IEEE Press, Piscataway, NJ, pp. 309–314, 1997.

147 T. Stützle and H. Hoos, *MAX–MIN Ant System*, Future Generation Computer Systems, vol. 16, no. 8, pp. 889–914, 2000.

148 S. Suliman, *A two-phase heuristic approach to the permutation flow-shop scheduling problem*, International Journal of Production Economics 64, 143–152, 2000.

149 Syswerda and J. Palmucci, *The Application of Genetic Algorithms to Resource Scheduling*, Proceedings of the Fourth International Conference on Genetic Algorithms, University of California, San Diego, Richard K. Belew and Lashon B. Booker, editors, pages 502-508, 13-16 July 1991.

150 V. T'kindt, N. Monmarche, F. Tercinet and D. Laügt*, An Ant Colony Optimization Algorithm to Solve a 2-machine Bicriteria Flowshop Scheduling Problem*, European Journal of Operational Research, 142:2, p. 250-257, 2002

151 E. Taillard, Some *efficient heuristic methods for the flow shop sequencing problem*, European Journal of Operational Research, 47, 67-74,1990

152 E. Taillard, 1993, *Benchmarks for basic scheduling problems*, European Journal of Operational Research 64,278–285, 1993.

153 S. Van Der Zwaan and C. Marques, *Ant colony optimisation for job shop scheduling*, Proceedings of the 3rd Workshop on Genetic Algorithms and Artificial Life, 1999.

154 A M. Widmer and A. Hertz, A *New Heuristic Method for the Flow Shop Sequencing Problem*, Euro. J. Opt. Res., 41, 186-193, 1989.

155 Wikipedia the free encyclopedia, *Kolmogorov-Smirnov test*, http://en.wikipedia.org/wiki/Kolmogorov-Smirnov_test.

156 E. Zitzler, S. Künzli, Indicator-*based selection in multiobjective search*, Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Birmingham, UK, 832–842, 2004

157 E. Zitzler, M. Laumanns, L. Thiele*, SPEA2: Improving the strength Pareto evolutionary algorithm*, Technical report 103, Computer Engineering and Networks Laboratory (TIK) Swiss Federal Institute of Technology, Zurich, Switzerland, May, 1998.

158 E. Zitzler and L. Thiele, *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Evolutionary Algorithm*, IEEE Transactions on Evolutionary Computation, 3(4):257-271, 1999.

159 E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca and V. Grunert da Fonseca, *Performance assessment of multiobjective optimizers: an analysis and review*, IEEE

Appendices

1 Single objective approach: makespan

For an algorithm using MMAS for the construction of the sequence, we have obtained the results presented in Table 15.

instances	*50x20*	*100x5*	*100x10*	*100x20*	*200x10*	*average*
p0=0						
I	0.0499	0.0000	0.0055	0.0549	0.0075	0.0235
T-I	0.0504	0.0000	0.0057	0.0545	0.0078	0.0237
T-E	0.0700	0.0013	0.0246	0.0775	0.0155	0.0378
I-E	0.0506	0.0000	0.0068	0.0586	0.0088	0.0250
E-I	0.0533	0.0000	0.0080	0.0616	0.0100	0.0260
T-I-E	0.0533	0.0000	0.0071	0.0578	0.0088	0.0254
T-E-I	0.0528	0.0000	0.0073	0.0599	0.0102	0.0260
p0=0.5						
I	0.0483	0.0000	0.0052	0.0542	0.0064	0.0228
T-I	0.0467	0.0000	0.0227	0.0532	0.0067	0.0259
T-E	0.0708	0.0005	0.0206	0.0781	0.0173	0.0375
I-E	0.0514	0.0000	0.0062	0.0578	0.0091	0.0249
E-I	0.0512	0.0000	0.0083	0.0585	0.0106	0.0257
T-I-E	0.0506	0.0000	0.0073	0.0562	0.0087	0.0246
T-E-I	0.0512	0.0000	0.0078	0.0604	0.0086	0.0256
p0=0.75						
I	0.0448	0.0000	0.0049	0.0509	0.0057	0.0213
T-I	0.0464	0.0000	0.0050	0.0521	0.0065	0.0220
T-E	0.0660	0.0004	0.0227	0.0753	0.0162	0.0361
I-E	0.3134	0.0000	0.0066	0.0545	0.0082	0.0271
E-I	0.0491	0.0000	0.0081	0.0567	0.0086	0.0245
T-I-E	0.0496	0.0000	0.0059	0.0558	0.0081	0.0272
T-E-I	0.0472	0.0000	0.0076	0.0578	0.0097	0.0245
p0=0.9						
I	0.0408	0.0000	0.0040	0.0468	0.0041	0.0192
T-I	0.0416	0.0000	0.0029	0.0477	0.0043	0.0193
T-E	0.0605	0.0004	0.0215	0.0711	0.0167	0.0340
I-E	0.0464	0.0000	0.0055	0.0539	0.0074	0.0226
E-I	0.0464	0.0000	0.0071	0.0563	0.0094	0.0238
T-I-E	0.0422	0.0000	0.0050	0.0526	0.0084	0.0216
T-E-I	0.0477	0.0000	0.0066	0.0576	0.0094	0.0243
p0=0.925						
I	0.0390	0.0000	0.0024	0.0455	0.0028	0.0179

T-I	0.0406	0.0000	0.0031	0.0462	0.0033	0.0186
T-E	0.0599	0.0002	0.0205	0.0716	0.0168	0.0338
I-E	0.0456	0.0000	0.0049	0.0509	0.0061	0.0215
E-I	0.0456	0.0000	0.0078	0.0554	0.0087	0.0235
T-I-E	0.0427	0.0000	0.0057	0.0511	0.0068	0.0213
T-E-I	0.0472	0.0000	0.0071	0.0550	0.0092	0.0237
p0=0.95						
I	**0.0371**	0.0000	0.0026	0.0436	0.0027	0.0172
T-I	0.0408	0.0000	0.0033	0.0437	0.0025	0.0180
T-E	0.0597	0.0002	0.0217	0.0732	0.0186	0.0347
I-E	0.0443	0.0000	0.0050	0.0508	0.0071	0.0214
E-I	0.0453	0.0000	0.0061	0.0539	0.0080	0.0227
p0=0.95	*50x20*	*100x5*	*100x10*	*100x20*	*200x10*	*average*
T-I-E	0.0435	0.0000	0.0043	0.0501	0.0069	0.0210
T-E-I	0.0448	0.0000	0.0064	0.0513	0.0086	0.0222
p0=0.975						
I	0.0411	0.0000	0.0024	0.0424	**0.0024**	0.0177
T-I	0.0400	0.0000	**0.0019**	**0.0409**	**0.0024**	**0.0171**
T-E	0.0607	0.0002	0.0205	0.0708	0.0163	0.0337
I-E	0.0435	0.0000	0.0054	0.0483	0.0061	0.0207
E-I	0.0451	0.0000	0.0068	0.0519	0.0085	0.0224
T-I-E	0.0451	0.0000	0.0045	0.0491	0.0084	0.1071
T-E-I	0.0464	0.0000	0.0054	0.0524	0.0083	0.1125

Table 15: Results achieved for the makespan using MMAS

We have put in bold the best values obtained for each instance. It appears clearly that I and T-I neighbourhood local search give the best results, independently of the number of jobs and the number of machines in the problem. This is due to the to the speedup technique of Taillard this technique can be used only when the objective to optimize is the makespan .Best performances are also obtained for a high value of p_0, $p_0 = 0.95$ or 0.975 what means that it is more relevant to focus on the exploitation of the results than on the exploration of the solution space if we want to find best solutions. The following Table 16 presents the results obtained with MMAS_sum for the makespan objective.

instances	50x20	100x5	100x10	100x20	200x10	average
p0=0						
I	0.04508	0	0.00364	0.046839	0.0035	0.0198
T-I	0.04429	0	0.003466	0.048641	0.00414	0.0201
T-E	0.06179	0.00018	0.018891	0.07075	0.01722	0.0338
I-E	0.048	0	0.005373	0.054209	0.007	0.0229
E-I	0.04853	0	0.006586	0.056502	0.00994	0.0235
T-I-E	0.04773	0	0.006412	0.052735	0.00644	0.0227
T-E-I	0.04959	0	0.023224	0.053881	0.0081	0.0270
p0=0.5						
I	0.04375	0	0.002946	0.046675	0.00221	0.0191
T-I	0.04375	0	0.002773	0.047494	0.00304	0.0194
T-E	0.05993	0.00018	0.016638	0.070423	0.01869	0.0332
I-E	0.04641	0	0.004853	0.051916	0.00562	0.0218
E-I	0.04694	0	0.006759	0.055355	0.00856	0.0235

T-I-E	0.048	0	0.005199	0.052407	0.00654	0.0224
T-E-I	0.04906	0	0.005893	0.052735	0.00792	0.0231
p0=0.75						
I	0.04216	0	0.002773	0.044219	0.00175	0.0182
T-I	0.04057	0	0.002253	0.045201	0.00175	0.0180
T-E	0.05914	0.00018	0.018198	0.071733	0.01897	0.0336
I-E	0.04375	0	0.004679	0.050115	0.00644	0.0241
E-I	0.0472	0	0.005893	0.052899	0.00829	0.0229
T-I-E	0.04508	0	0.005026	0.050115	0.00709	0.0243
T-E-I	0.0472	0	0.006932	0.054209	0.00847	0.0234
p0=0.9	50x20	100x5	100x10	100x20	200x10	average
I	**0.0366**	0	**0.001906**	0.042909	0.00212	0.0167
T-I	0.03872	0	0.00312	0.041598	0.00193	0.0171
T-E	0.04375	0.00036	0.019757	0.070259	0.01915	0.0307
I-E	0.04296	0	0.005199	0.048804	0.0058	0.0206
E-I	0.04322	0	0.007972	0.051752	0.00856	0.0223
T-I-E	0.04349	0	0.005373	0.048968	0.00626	0.0208
T-E-I	0.04455	0	0.006066	0.054045	0.00948	0.0228
p0=0.925						
I	0.03739	0	0.00208	0.039469	**0.00166**	**0.0161**
T-I	0.03898	0	**0.001906**	**0.039142**	0.00193	0.0164
T-E	0.06152	0.00073	0.019237	0.071405	0.01749	0.0341
I-E	0.04375	0	0.00364	0.047003	0.00552	0.0200
E-I	0.04535	0	0.006412	0.05339	0.00911	0.0229
T-I-E	0.04482	0	0.004506	0.050115	0.00672	0.0212
T-E-I	0.04322	0	0.006239	0.051752	0.00994	0.0222
p0=0.95						
I	0.03898	0	0.002946	0.042417	0.00193	0.0173
T-I	0.03872	0	0.002946	0.039961	**0.00166**	0.0167
T-E	0.0602	0.00073	0.021144	0.075336	0.01786	0.0351
I-E	0.04322	0	0.004506	0.046839	0.00635	0.0202
E-I	0.04588	0	0.006586	0.050442	0.00635	0.0219
T-I-E	0.04588	0	0.004853	0.049296	0.00626	0.0213
T-E-I	0.04667	0	0.006239	0.052244	0.0069	0.0224
p0=0.975						
I	0.03951	0	0.002773	0.040616	0.00285	0.0172
T-I	0.03872	0	0.00312	**0.039142**	0.00184	0.0166
T-E	0.06391	0.00164	0.022357	0.071896	0.01924	0.0358
I-E	0.04535	0	0.005026	0.04471	0.00792	0.0206
E-I	0.04747	0	0.005893	0.05208	0.0081	0.0227
T-I-E	0.04535	0	0.005373	0.047658	0.00635	0.0209
T-E-I	0.04667	0	0.008839	0.051752	0.00746	0.0229

Table 16: Results achieved for the makespan using MMAS +sum

As for the previous table, best results for each instance are put in bold. The results obtained with MMAS_sum for the construction of the solution give the same observations; better results are obtained when using I or T-I and this for high value of p_0, between 0.9 and 0.975.

Another interesting result is that the using MMAS_sum gives better results than using MMAS, and that for all the instances tested.

2 Single objective approach: Total tardiness

For the total tardiness the results obtained with MMAS and with MMAS_sum are presented respectively in the Tables 17 and 18.

instance	50x10	50x50	average
p0=0			
I	0.16692587	0.10080483	0.13386535
E	0.22083981	0.15320494	0.18702238
T-E	0.22083981	0.14561656	0.18322818
T-I	0.15914982	0.09563093	0.12739037
I-E	0.17366511	0.11066398	0.14216455
E-I	0.15811301	0.09617706	0.12714504
T-E-I	0.15863142	0.10008623	0.12935882
T-I-E	0.15966822	0.09752803	0.12859812
p0=0.5			
I	0.14307932	0.09453866	0.11880899
E	0.24157595	0.13972406	0.19065
T-E	0.20165889	0.14133372	0.1714963
T-I	0.12649041	0.09959759	0.113044
I-E	0.16174184	0.08870365	0.12522274
E-I	0.15863142	0.08709399	0.1228627
T-E-I	0.12078797	0.0915493	0.10616863
T-I-E	0.12649041	0.08904858	0.10776949
p0=0.75			
I	0.12804562	0.09537223	0.11170893
E	0.18403318	0.12880138	0.15641728
T-E	0.18040435	0.13342915	0.15691675
T-I	0.08553655	0.07663122	0.08108388
I-E	0.13219285	0.0837597	0.10797627
E-I	0.12078797	0.07803967	0.09941382
T-E-I	0.09590461	0.08169014	0.08879738
T-I-E	0.08761016	0.07344064	0.0805254
p0=0.9			
I	0.10782789	0.08450704	0.09616747
E	0.17314671	0.1221328	0.14763975
T-E	0.11145671	0.11345214	0.11245443
T-I	0.07672369	0.06622593	0.07147481
I-E	0.09901503	0.08950848	0.09426176
E-I	0.09590461	0.07700489	0.08645475
T-E-I	0.0627268	0.06754815	0.06513747
T-I-E	0.05132193	0.07355562	0.06243877
p0=0.925			
I	0.1088647	0.09678068	0.10282269
E	0.15448419	0.14173613	0.14811016
T-E	0.12856402	0.1350388	0.13180141
T-I	0.05495075	**0.06642713**	0.**06068894**
I-E	0.10419907	0.10250072	0.10334989
E-I	0.0627268	0.07677494	0.06975087
T-E-I	0.07568688	0.07174475	0.07371582
T-I-E	**0.03939865**	0.07436045	0.05687955
p0=0.95			
I	0.1093831	0.10068985	0.10503648

E	0.19388284	0.14443806	0.16916045
T-E	0.12182478	0.16047715	0.14115096
T-I	0.04510109	0.07079621	0.05794865
I-E	0.11664075	0.09867778	0.10765926
E-I	0.07568688	0.0731532	0.07442004
0.95			
T-E-I	0.05702436	0.09617706	0.07660071
T-I-E	0.06117159	0.08180512	0.07148835
p0=0.975			
I	0.093831	0.10385168	0.09884134
E	0.24779679	0.16223053	0.20501366
T-E	0.12234318	0.15791894	0.14013106
T-I	0.0632452	0.09430871	0.07877696
I-E	0.12597201	0.10689853	0.11643527
E-I	0.07257646	0.10192584	0.08725115
T-E-I	0.07309487	0.10296062	0.08802774
T-I-E	0.08864697	0.10103478	0.09484087

Table 17: Results achieved for the total tardiness when using MMAS

We have put in bold the best values obtained for each instance. By contrast with the results for the makespan, it is not clear whether one configuration of local search is better than another but we can still observe that best results are still obtained for high value of p_0

instances	50x10	50x50	average
p0=0			
I	0.09642302	0.07390055	0.08516178
E	0.16796267	0.12569704	0.14682986
T-E	0.13893209	0.11925841	0.12909525
T-I	0.08294453	0.07378557	0.07836505
I-E	0.10212545	0.0769474	0.08953643
E-I	0.08346293	0.06223053	0.07284673
T-E-I	0.07309487	0.07045128	0.07177307
T-I-E	0.08138932	0.07346939	0.07742935
p0=0.5			
I	0.06635562	0.06501868	0.06568715
E	0.12337999	0.12219028	0.12278514
T-E	0.10368066	0.11713136	0.11040601
T-I	0.07205806	0.06438632	0.06822219
I-E	0.06998445	0.08192009	0.07595227
E-I	0.06894764	0.06021845	0.06458305
T-E-I	0.06998445	0.05651049	0.06324747
T-I-E	0.07361327	0.05757402	0.06559364
p0=0.75			
I	0.0627268	0.06674332	0.06473506
E	0.08761016	0.11279103	0.1002006
T-E	0.07983411	0.11029031	0.09506221
T-I	0.04458269	0.04751365	0.04604817
I-E	0.0632452	0.06521989	0.06423255
	50x10	50x50	average
E-I	0.05132193	0.05412475	0.05272334
T-E-I	0.05495075	0.05145157	0.05320116
T-I-E	0.05184033	0.05403852	0.05293942

p0=0.9			
I	0.05754277	0.065881	0.06171188
E	0.08501814	0.10143719	0.09322767
T-E	0.0627268	0.09137683	0.07705182
T-I	0.05391395	0.**04943949**	0.05167672
p0=0.9	50x10	50x50	average
I-E	0.07102125	0.06504743	0.06803434
E-I	0.06117159	0.06263294	0.06190227
p0=9.0	50x10	50x50	average
T-E-I	0.06168999	0.05817764	0.05993382
T-I-E	0.05287714	0.05768899	0.05528306
p0=0.925			
I	0.05806117	0.06113826	0.05959971
E	0.07724209	0.11943087	0.09833648
T-E	0.06791083	0.09902271	0.08346677
T-I	0.03680664	0.0531762	0.04499142
I-E	0.06168999	0.06611095	0.06390047
E-I	0.**03265941**	0.05251509	0.**04258725**
T-E-I	0.05339554	0.06286289	0.05812922
T-I-E	0.04354588	0.0597873	0.05166659
p0=0.95			
I	0.04354588	0.06036217	0.05195403
E	0.08242613	0.10750216	0.09496414
T-E	0.07257646	0.11509054	0.0938335
T-I	0.03888025	0.0522564	0.04556832
I-E	0.07568688	0.07355562	0.07462125
E-I	0.05650596	0.05941362	0.05795979
T-E-I	0.05754277	0.06131072	0.05942674
T-I-E	0.06480041	0.07298074	0.06889058
p0=0.975			
I	0.09953344	0.07565392	0.08759368
E	0.17366511	0.11279103	0.14322807
T-E	0.06117159	0.13705088	0.09911123
T-I	0.07724209	0.1054326	0.09133734
I-E	0.1254536	0.07085369	0.09815365
E-I	0.0782789	0.08378844	0.08103367
T-E-I	0.07413167	0.09315895	0.08364531
T-I-E	0.09486781	0.08051164	0.08768972

Table 18: Results achieved for the total tardiness when using MMAS_sum

Best results are still obtained for a high value of p_0 and when MMAS_sum is used. For the local search. none of them seems to perform better than the others. The only conclusion we can make on the local search for the total tardiness is that it includes insert neighbourhood.

3 Outcomes of multiobjective optimizer for different local search

For these comparisons, we have run *1phero scratch* and *1phero 2phase* with the different combinations of local search presented in the section 5.2. For each configuration, five independent runs have been achieved on 50x10-1 briefly described in section 5.4.3, and this for a number of aggregation weight $|W| = 41$ and a direction changes $\lambda = 1...0$

In the tables 19 and 20, the values of the different values of quality indicators are presented for each configuration, first for *1phero scratch* and then for *1phero 2phase.*

$I_C(I,T-I)$	54%	$I_C(T-I,I)$	43%
$I_H(I)$	3.0515E+08	$I_H(T-I)$	3.0480E+08
$I_{H2}(I,T-I)$	18106.7	$I_{H2}(T-I,I)$	4225.21
$I_C(I,EI)$	47%	$I_C(E-I,I)$	50%
$I_H(I)$	3.0515E+08	$I_H(E-I)$	3.0471E+08
$I_{H2}(I,E-I)$	22037.4	$I_{H2}(E-I,I)$	4696.7
$I_C(I,I-E)$	58%	$I_C(I-E,I)$	37%
S(I)	3.0515E+08	$I_H(I-E)$	3.0521E+08
$I_{H2}(I,I-E)$	1188.8	$I_{H2}(I-E,I)$	13981.1
$I_C(I,T-I-E)$	57%	$I_C(T-I-E,I)$	38%
$I_H(I)$	3.0515E+08	S($I_H(T-I-E)$)	3.0478E+08
$I_{H2}(I,T-I-E)$	20604.3	$I_{H2}(T-I-E,I)$	5971.92
$I_C(I,T-E-I)$	50%	$I_C(T-E-I,I)$	43%
$I_H(I)$	3.0515E+08	$I_H(T-E-I)$	3.0489E+08
$I_{H2}(I,T-E-I)$	16320.1	$I_{H2}(T-E-I,I)$	6024.07
$I_C(T-I,E-I)$	45%	$I_C(E-I,T-I)$	49%
$I_H(T-I)$	3.0480E+08	$I_H(E-I)$	3.0471E+08
$I_{H2}(T-I,E-I)$	9330.32	$I_{H2}(E-I,T-I)$	5871.16
$I_C(T-I,I-E)$	57%	$I_C(I-E,T-I)$	27%
$I_H(T-I)$	3.0480E+08	$I_H(I-E)$	3.0521E+08
$I_{H2}(T-I,I-E)$	3002.57	$I_{H2}(I-E,T-I)$	19432
$I_C(T-I,T-E-I)$	43%	$I_C(T-E-I,T-I)$	46%
$I_H(T-I)$	3.0480E+08	$I_H(T-E-I)$	3.0489E+08
$I_{H2}(T-I,T-E-I)$	4273.54	$I_{H2}(T-E-I,T-I)$	7854.86
$I_C(T-I,T-I-E)$	56%	$I_C(T-I-E,T-I)$	38%
$I_H(T-I)$	3.0480E+08	$I_H(T-I-E)$	3.0478E+08
$I_{H2}(T-I,T-I-E)$	9037.08	$I_{H2}(T-I-E,T-I)$	8286.23
$I_C(E-I,I-E)$	56%	$I_C(I-E,E-I)$	36%
$I_H(E-I)$	3.0471E+08	$I_H(I-E)$	3.0521E+08

$I_{H2}(E-I,I-E)$	3316.4	$I_{H2}(I-E,E-I)$	23205
$I_C(E-I,T-I-E)$	58%	$I_C(T-I-E,E-I)$	36%
$I_H(E-I)$	3.0471E+08	$I_H(T-I-E)$	3.0478E+08
$I_{H2}(E-I,T-I-E)$	8436.01	$I_{H2}(T-I-E,E-I)$	11144.3
$I_C(E-I,T-E-I)$	45%	$I_C(T-E-I,E-I)$	48%
$I_H(E-I)$	3.0471E+08	$I_H(T-E-I)$	3.0489E+08
$I_{H2}(E-I,T-E-I)$	4703.95	$I_{H2}(T-E-I,E-I)$	11745.4
$I_C(I-E,T-I-E)$	34%	$I_C(T-I-E,I-E)$	50%
$I_H(I-E)$	3.0521E+08	$I_H(T-I-E)$	3.0478E+08
D(IE.TIE)	21710.3	$I_{H2}(T-I-E,I-E)$	4530.01
$I_C(I-E,T-E-I)$	33%	$I_C(T-E-I,I-E)$	55%
$I_H(I-E)$	3.0521E+08	$I_H(T-E-I)$	3.0489E+08
$I_{H2}(I-E,T-E-I)$	16357.9	$I_{H2}(E-I,T-I)$	3519.36
$I_C(T-E-I,T-I-E)$	58%	$I_C(T-I-E,T-E-I)$	35%
$I_H(T-E-I)$	3.0489E+08	$I_H(T-I-E)$	3.0478E+08
$I_{H2}(T-E-I,T-I-E)$	11068.1	$I_{H2}(E-I-E,T-E-I)$	6736.4

Table 19:Results for *1phero scratch* for different local search

1phero 2phase

$I_C(I,T-I)$	55%	$I_C(T-I,I)$	30%
$I_H(I)$	3.0563E+08	$I_H(T-I)$	3.0598E+08
$I_{H2}(I,T-I)$	9048	$I_{H2}(T-I,I)$	23036.7
$I_C(I,EI)$	61%	$I_C(E-I,I)$	29%
$I_H(I)$	3.0563E+08	$I_H(E-I)$	3.0487E+08
$I_{H2}(I,E-I)$	33426.5	$I_{H2}(E-I,I)$	2836.43
$I_C(I,I-E)$	64%	$I_C(I-E,I)$	23%
S(I)	3.0563E+08	$I_H(I-E)$	3.0528E+08
$I_{H2}(I,I-E)$	18857.8	$I_{H2}(I-E,I)$	4668.34
$I_C(I,T-I-E)$	58%	$I_C(T-I-E,I)$	30%
$I_H(I)$	3.0563E+08	S($I_H(T-I-E)$)	3.0469E+08
$I_{H2}(I,T-I-E)$	37817.4	$I_{H2}(T-I-E,I)$	200.606
$I_C(I,T-E-I)$	55%	$I_C(T-E-I,I)$	38%
$I_H(I)$	3.0563E+08	$I_H(T-E-I)$	3.0527E+08
$I_{H2}(I,T-E-I)$	23576.8	$I_{H2}(T-E-I,I)$	9003.31
$I_C(T-I,E-I)$	40%	$I_C(E-I,T-I)$	42%
$I_H(T-I)$	3.0563E+08	$I_H(E-I)$	3.0487E+08
$I_{H2}(T-I,E-I)$	48338.1	$I_{H2}(E-I,T-I)$	3759.22
$I_C(T-I,I-E)$	43%	$I_C(I-E,T-I)$	46%
$I_H(T-I)$	3.0598E+08	$I_H(I-E)$	3.0528E+08
$I_{H2}(T-I,I-E)$	33326.2	$I_{H2}(I-E,T-I)$	5148.04
$I_C(T-I,T-E-I)$	32%	$I_C(T-E-I,T-I)$	0.524159
$I_H(T-I)$	3.0563E+08	$I_H(T-E-I)$	3.0527E+08
$I_{H2}(T-I,T-E-I)$	36582.5	$I_{H2}(T-E-I,T-I)$	8019.8
$I_C(T-I,T-I-E)$	37%	$I_C(T-I-E,T-I)$	47%
$I_H(T-I)$	3.0598E+08	$I_H(T-I-E)$	3.0469E+08

$I_{H2}(T-I,T-I-E)$	53379	$I_{H2}(T-I-E,T-I)$	1773.47
$I_C(E-I,I-E)$	42%	$I_C(I-E,E-I)$	46%
$I_H(E-I)$	3.0487E+08	$I_H(I-E)$	3.0528E+08
$I_{H2}(E-I,I-E)$	7133.73	$I_{H2}(I-E,E-I)$	23534.4
$I_C(E-I,T-I-E)$	44%	$I_C(T-I-E,E-I)$	49%
$I_H(E-I)$	3.0487E+08	$I_H(T-I-E)$	3.0469E+08
$I_{H2}(E-I,T-I-E)$	15791.3	$I_{H2}(T-I-E,E-I)$	8764.69
$I_C(E-I,T-E-I)$	42%	$I_C(T-E-I,E-I)$	54%
$I_H(E-I)$	3.0487E+08	$I_H(T-E-I)$	3.0527E+08
$I_{H2}(E-I,T-E-I)$	9088.75	$I_{H2}(T-E-I,E-I)$	25105.7
$I_C(I-E,T-I-E)$	42%	$I_C(T-I-E,I-E)$	49%
$I_H(I-E)$	3.0528E+08	$I_H(T-I-E)$	3.0469E+08
D(IE.TIE)	25582.7	$I_{H2}(T-I-E,I-E)$	2155.37
$I_C(I-E,T-E-I)$	37%	$I_C(T-E-I,I-E)$	55%
$I_H(I-E)$	3.0528E+08	$I_H(T-E-I)$	3.0527E+08
$I_{H2}(I-E,T-E-I)$	13882	$I_{H2}(E-I,T-I)$	13497.5
$I_C(T-E-I,T-I-E)$	53%	$I_C(T-I-E,T-E-I)$	48%
$I_H(T-E-I)$	3.0527E+08	$I_H(T-I-E)$	3.0469E+08
$I_{H2}(T-E-I,T-I-E)$	26267.2	$I_{H2}(E-I-E,T-E-I)$	3224.39

Table 20: Results for *1phero 2phase* for different local search

The result of these tests is that no combination of local search outperforms another. This result was expected as no combinations of local search had clearly shown better performance for the total tardiness objective in the single objective approach.

4 Influence of the number of aggregations weights

We will present here the results of the comparisons concerning the influence of the number of aggregations weights on the performance of the optimizer. The different configurations have been tested on the second and 50x30-2 for a direction changes of type $\lambda = 1...0$. Table 21 shows the coverage measure and the result of the K-S test for the different comparisons made with *1phero 2phase* for 50x10-2.

***1phero 2phase* - λ = 1...0 - (50x10-2)**	I_C	**Kolmogorov-Smirnov test**
$(1phero\ 2phase - \lvert W \rvert = 11), (1phero\ 2phase - \lvert W \rvert = 41)$	25%	The two attainment surfaces differ somewhere
$(1phero\ 2phase - \lvert W \rvert = 41), (1phero\ 2phase - \lvert W \rvert = 11)$	65%	
$(1phero\ 2phase - \lvert W \rvert = 11), (1phero\ 2phase - \lvert W \rvert = 81)$	28%	The two attainment surfaces differ somewhere
$(1phero\ 2phase - \lvert W \rvert = 81), (1phero\ 2phase - \lvert W \rvert = 11)$	62%	
$(1phero\ 2phase - \lvert W \rvert = 41), (1phero\ 2phase - \lvert W \rvert = 81)$	49%	h_0 not rejected
$(1phero\ 2phase - \lvert W \rvert = 81), (1phero\ 2phase - \lvert W \rvert = 41)$	42%	

Table 21: Results for different numbers of weights for *1phero 2phase* (50x10-2)

We can observe that no significant differences exist between the EAFs of the configurations 1phero 2phase with $\lvert W \rvert = 41$ and $\lvert W \rvert = 81$. The two Figures 43 and 44 indicate that a number of weights $\lvert W \rvert = 41$ is not sufficient to find solutions of good quality.

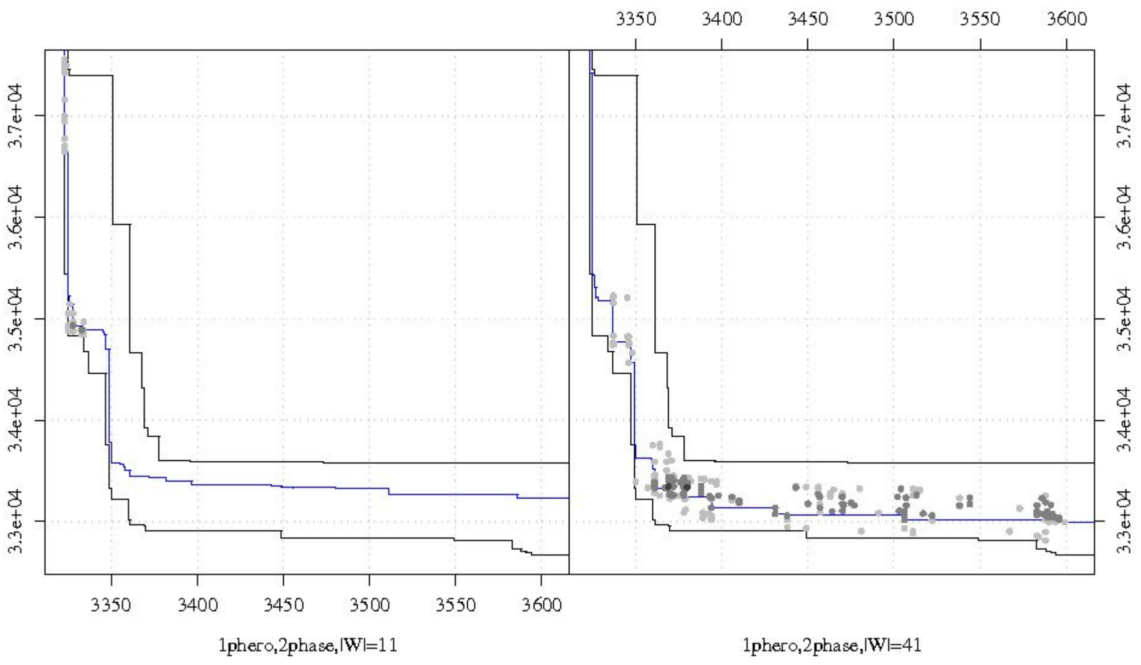


Figure 43: Differences of EAFs, *1phero 2phase* |W|=11-|W|=41 (50x10-2)

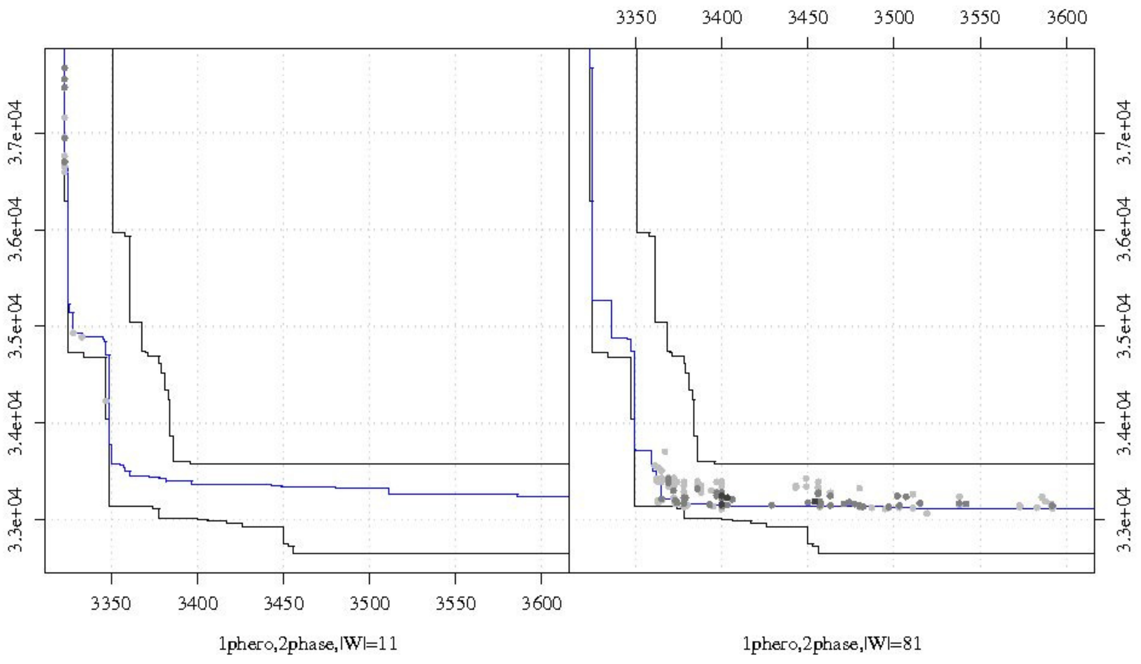


Figure 44: Differences of EAFs, *1phero 2phase* |W|=11-|W|=81 (50x10-2)

Table 22 and 23 shows the coverage measure and the result of the K-S test for the different comparisons made with respectively *2pheroG 2phase* and *2pheroL 2phase. They show* that the number of aggregation weights does not have much influence on the performance of the configurations *2pheroG 2phase* and *2pheroL 2phase*. Table 24 shows that a comparison based on the plot of differences of EAFs is possible for the comparisons made with 1phero 2phase on 50x30-2 is possible, except for a comparison $|W| = 41$ - $|W| = 81$

2pheroG 2phase- $\lambda = 1$□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
$(2pheroG\ \text{2phase} - \lvert W \rvert = 11), (2pheroG\ \text{2phase} - \lvert W \rvert = 41)$	42%	h_0 not rejected
$(2pheroG\ \text{2phase} - \lvert W \rvert = 41), (2pheroG\ \text{2phase} - \lvert W \rvert = 11)$	48%	
$(2pheroG\ \text{2phase} - \lvert W \rvert = 11), (2pheroG\ \text{2phase} - \lvert W \rvert = 81)$	41%	h_0 not rejected
$(2pheroG\ \text{2phase} - \lvert W \rvert = 81), (2pheroG\ \text{2phase} - \lvert W \rvert = 11)$	49%	
$(2pheroG\ \text{2phase} - \lvert W \rvert = 41), (2pheroG\ \text{2phase} - \lvert W \rvert = 81)$	50%	h_0 not rejected
$(2pheroG\ \text{scratch} - \lvert W \rvert = 81), (2pheroG\ \text{scratch} - \lvert W \rvert = 41)$	43%	

Table 22: Results for different number of weights for *2pheroG 2phase* (50x10-2)

2pheroL 2phase- $\lambda = 1$□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
$(2pheroL\ \text{2phase} - \lvert W \rvert = 11), (2pheroL\ \text{2phase} - \lvert W \rvert = 41)$	48%	h_0 not rejected
$(2pheroL\ \text{2phase} - \lvert W \rvert = 41), (2pheroL\ \text{2phase} - \lvert W \rvert = 11)$	49%	
$(2pheroL\ \text{2phase} - \lvert W \rvert = 11), (2pheroL\ \text{2phase} - \lvert W \rvert = 81)$	37%	h_0 not rejected
$(2pheroL\ \text{2phase} - \lvert W \rvert = 81), (2pheroL\ \text{2phase} - \lvert W \rvert = 11)$	56%	
$(2pheroL\ \text{2phase} - \lvert W \rvert = 41), (2pheroL\ \text{2phase} - \lvert W \rvert = 81)$	42%	h_0 not rejected
$(2pheroL\ \text{scratch} - \lvert W \rvert = 81), (2pheroL\ \text{scratch} - \lvert W \rvert = 41)$	52%	

Table 23: Results for different number of weights for *2pheroL (*50x10-2)

1phero 2phase - $\lambda = 1$□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
$(1phero\ 2phase - \lvert W \rvert = 11), (1phero\ 2phase - \lvert W \rvert = 41)$	24%	The two attainment surfaces differ somewhere
$(1phero\ 2phase - \lvert W \rvert = 41), (1phero\ 2phase - \lvert W \rvert = 11)$	67%	
$(1phero\ 2phase - \lvert W \rvert = 11), (1phero\ 2phase - \lvert W \rvert = 81)$	26%	The two attainment surfaces differ somewhere
$(1phero\ 2phase - \lvert W \rvert = 81), (1phero\ 2phase - \lvert W \rvert = 11)$	64%	
$(1phero\ 2phase - \lvert W \rvert = 41), (1phero\ 2phase - \lvert W \rvert = 81)$	43%	h_0 not rejected
$(1phero\ 2phase - \lvert W \rvert = 81), (1phero\ 2phase - \lvert W \rvert = 41)$	47%	

Table 24: Results for different number of weights for *1phero 2phase* (50x30-2)

The two Figures 45 and 46 provide the same kind of observations. Increasing the number of aggregation weights from $|W| = 11$ to $|W| = 41$ or $|W| = 81$ improves the performance of the configuration *1phero 2phase* in finding solutions in the middle region of the objective space.

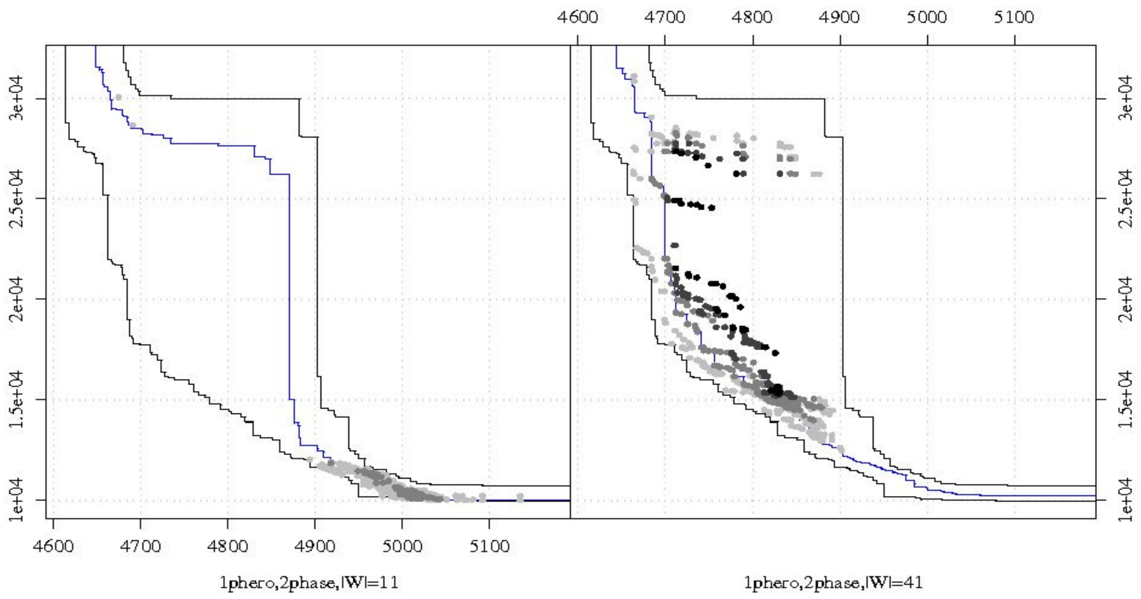


Figure 45: Differences of EAFs, *1phero 2phase* |W|=11-|W|=41 (50x30-2)

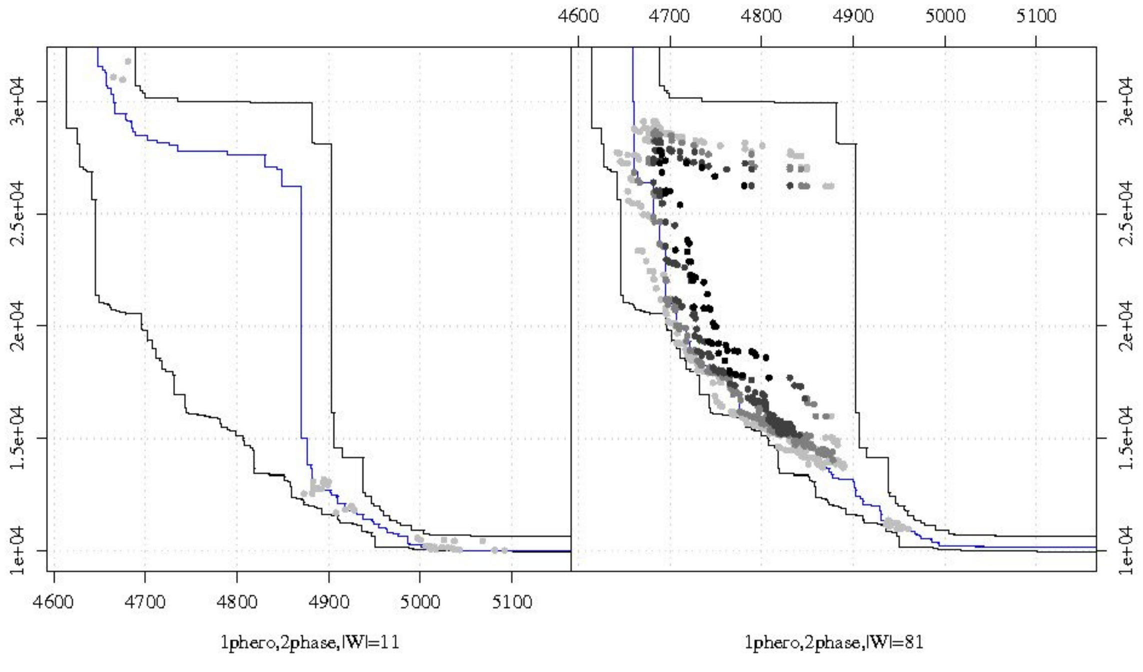


Figure 46: Differences of EAFs, *1phero 2phase* |W|=11-|W|=81 (50x30-2)

The Table 25 and 26 also show that no dominance relation appears, but plot of the differences of EAFs can be used for comparison for the tests made with respectively *2pheroG 2phase* and *2pheroL 2phase* on 50x30-2.

2pheroG 2phase $\lambda = 1$□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
$(2\,pheroG\ \text{2phase} - \lvert W \rvert = 11), (2\,pheroG\ \text{2phase} - \lvert W \rvert = 41)$	23%	The two attainment surfaces differ somewhere
$(2\,pheroG\ \text{2phase} - \lvert W \rvert = 41), (2\,pheroG\ \text{2phase} - \lvert W \rvert = 11)$	69%	
$(2\,pheroG\ \text{2phase} - \lvert W \rvert = 11), (2\,pheroG\ \text{2phase} - \lvert W \rvert = 81)$	25%	The two attainment surfaces differ somewhere
$(2\,pheroG\ \text{2phase} - \lvert W \rvert = 81), (2\,pheroG\ \text{2phase} - \lvert W \rvert = 11)$	63%	
$(2\,pheroG\ \text{2phase} - \lvert W \rvert = 41), (2\,pheroG\ \text{2phase} - \lvert W \rvert = 81)$	41%	The two attainment surfaces differ somewhere
$(2\,pheroG\ \text{scratch} - \lvert W \rvert = 81), (2\,pheroG\ \text{scratch} - \lvert W \rvert = 41)$	49%	

Table 25: Results for different number of weights for *2pheroG 2phase* (50x30-2)

Figures 47, 48, 50 and 51 provide the same kind of observations, increasing the number of aggregation weights from $\lvert W \rvert = 11$ to $\lvert W \rvert = 41$ or $\lvert W \rvert = 81$ clearly improves the performance of the configuration *2pheroL 2phase* in finding solutions in the middle region of the objective space. Figures 49 and 52 indicate that after a certain number of weights, increasing the number does not lead anymore to an improvement of the performance in the middle region. This number depends probably on the instance and the kind of problem.

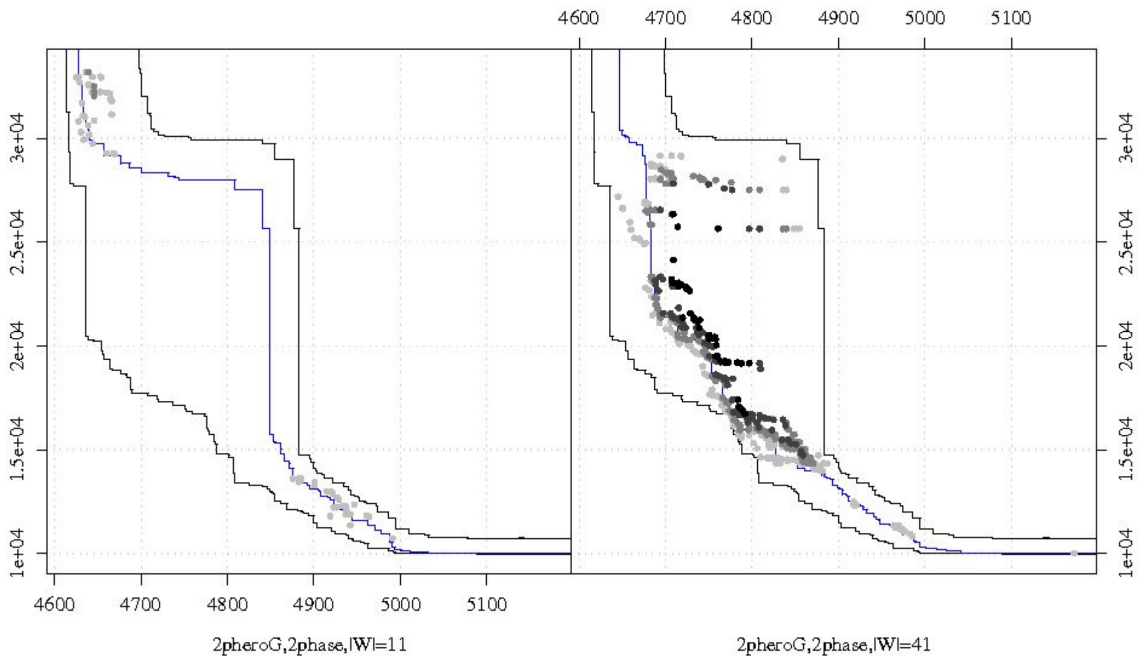


Figure 47: Differences of EAFs, *2pheroG 2phase* |W|=11-|W|=41 (50x30-2)

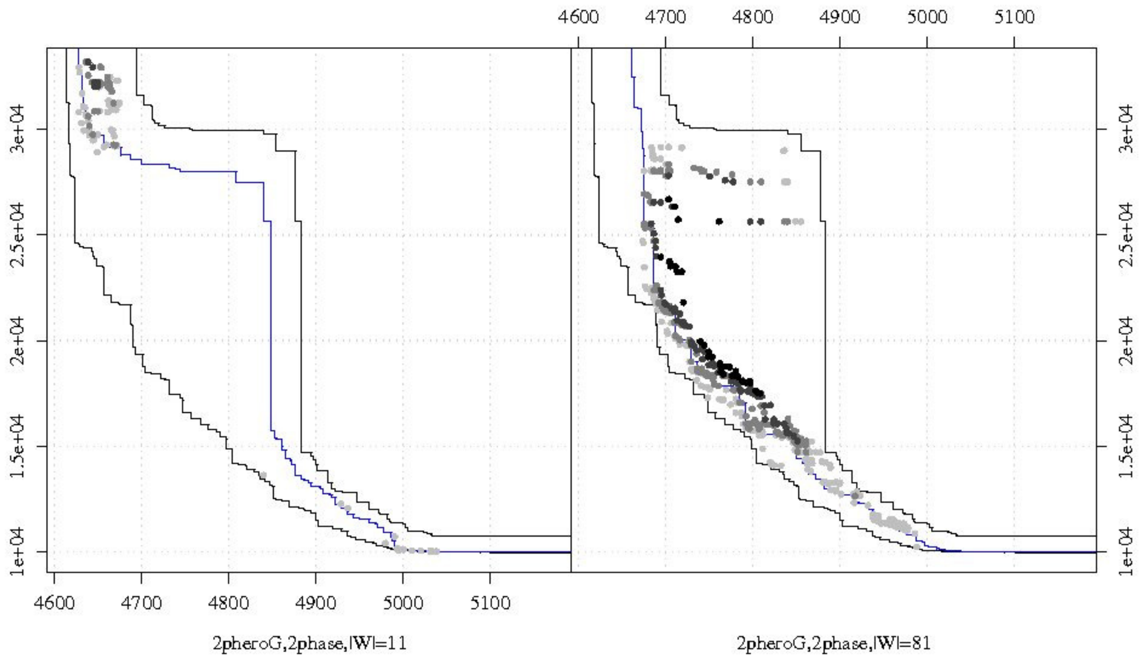


Figure 48: Differences of EAFs, *2pheroG 2phase* |W|=11-|W|=81 (50x30-2)

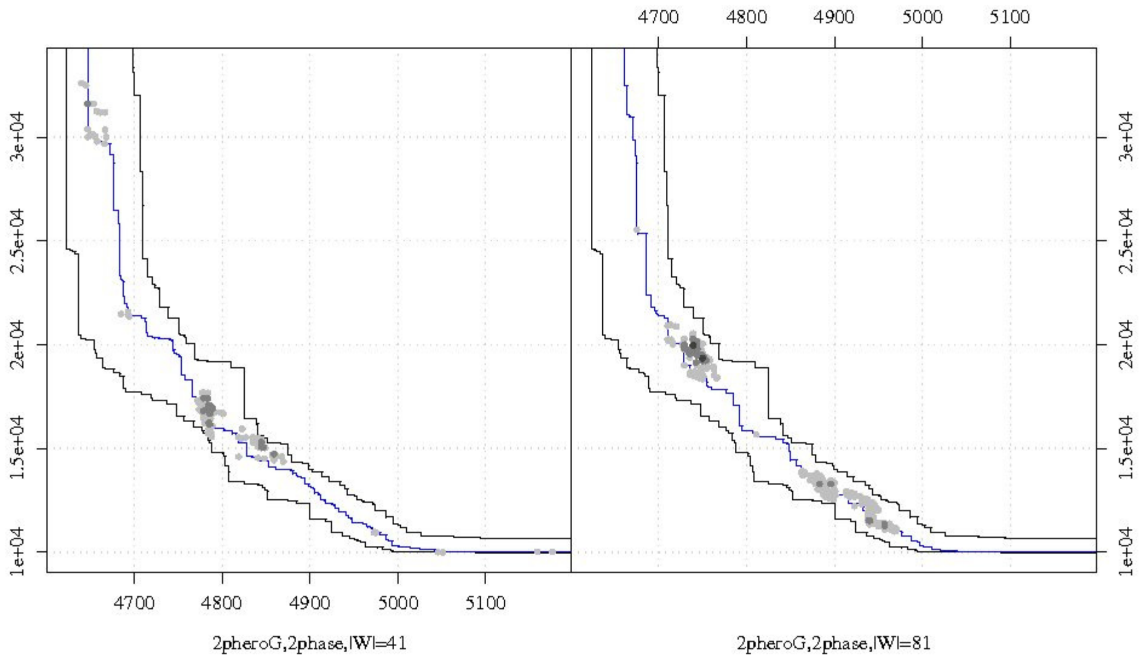


Figure 49: Differences of EAFs, *2pheroG 2phase* |W|=41-|W=81| (50x30-2)

1phero 2phase - $\lambda = 1$□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
$(2\,pheroL\ \text{2phase} - \lvert W \rvert = 11), (2\,pheroL\ \text{2phase} - \lvert W \rvert = 41)$	33%	The two attainment surfaces differ somewhere
$(2\,pheroL\ \text{2phase} - \lvert W \rvert = 41), (2\,pheroL\ \text{2phase} - \lvert W \rvert = 11)$	65%	
$(2\,pheroL\ \text{2phase} - \lvert W \rvert = 11), (2\,pheroL\ \text{2phase} - \lvert W \rvert = 81)$	34%	The two attainment surfaces differ somewhere
$(2\,pheroL\ \text{2phase} - \lvert W \rvert = 81), (2\,pheroL\ \text{2phase} - \lvert W \rvert = 11)$	55%	
$(2\,pheroL\ \text{2phase} - \lvert W \rvert = 41), (2\,pheroL\ \text{2phase} - \lvert W \rvert = 81)$	34%	The two attainment surfaces differ somewhere
$(2\,pheroL\ \text{scratch} - \lvert W \rvert = 81), (2\,pheroL\ \text{scratch} - \lvert W \rvert = 41)$	54%	

Table 26: Results for different number of weights for *2pheroL 2phase* (50x30-2)

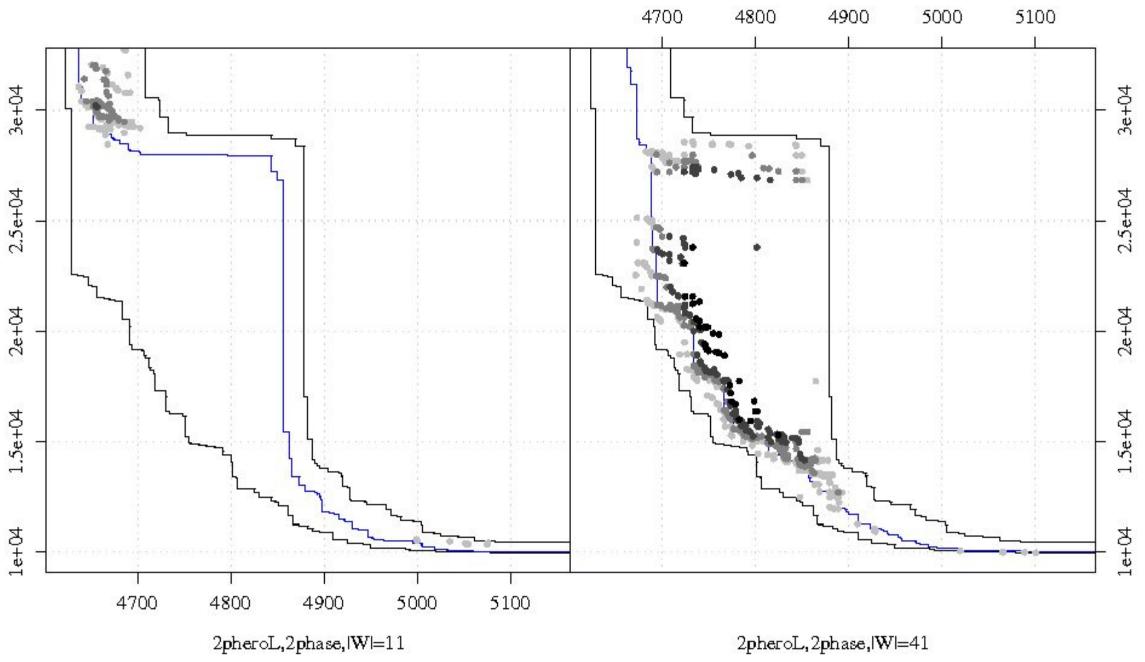


Figure 50: Differences of EAFs, *2pheroL 2phase* |W|=11-|W|=41 (50x30-2)

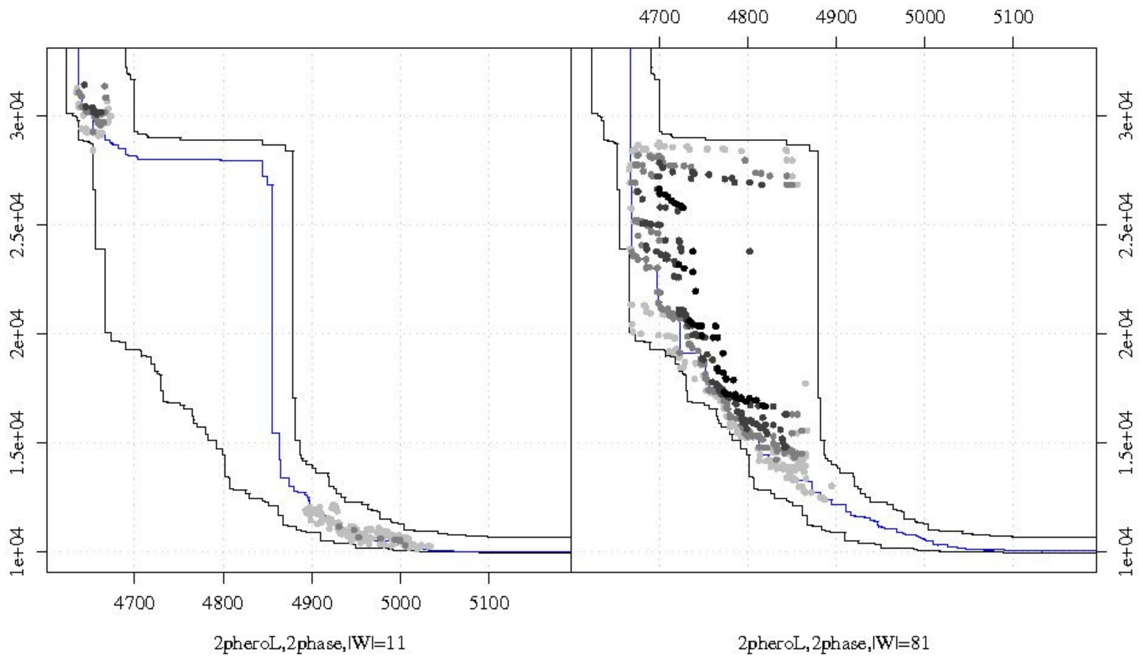


Figure 51: Differences of EAFs, *2pheroL 2phase* |W|=11-|W|=81| (50x30-2)

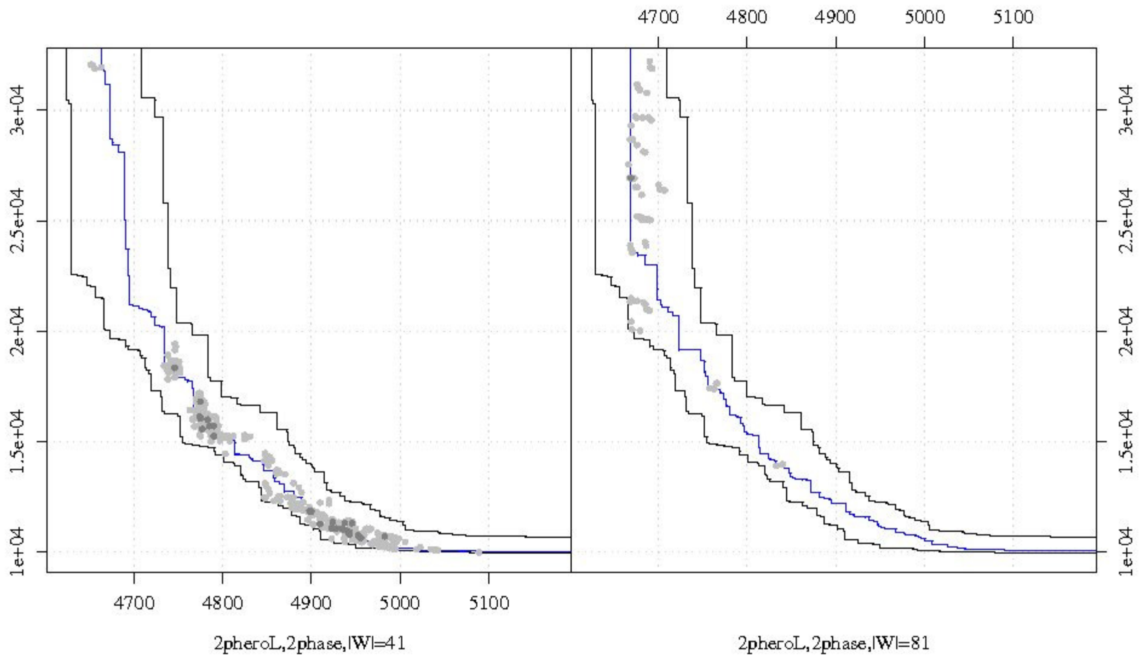


Figure 52: Differences of EAFs, *2pheroL* 2phase |W|=41-|W|=81 (50x30-2)

Summary of the observations of the influence of the number of aggregation weights

- Increasing the number of weights from $|W| = 11$ to $|W| = 41$ or to $|W| = 81$ does not lead to worse performance. Most of time the results are positive especially for the configuration *1phero 2 phase* and the 50 jobs 30 machines instance tested. For this instance, we have observed the necessity of using a high number of weights.. Otherwise, the algorithm is not capable to find solutions in the middle of the front. The fact that performance seems better at the extreme when using $|W| = 11$ shows that better solutions could be obtained if more time was allocated to the procedure.

- Increasing the number of weights from $|W| = 41$ to $|W| = 81$ ever leads to a clear improvement for the instances tested, what suggests that after a certain number of weights, increasing the number of aggregation weights does not lead anymore to an improvement of the performance.

5 Results of the optimizers for different direction changes

For these comparisons, we have run *1phero 2phase, 21pheroG 2phase and 2pheroL 2phase* with the different types of direction changes presented in section 5.4.5. For each configuration, ten independent runs have been achieved with a number of weights $|W| = 41$, on the second and 50x30-2 briefly described in section 5.4.3.

In the Tables 27, 28 and 29, we present the result of the tests made on 50x10-2.

Some situations of dominance appear in table 29 when the *configuration 2pheroL 2phase* is used. The coverage measures indicate that a direction changes of type $\lambda = 1$□ 0 is worse performing than the others.

For the two other configurations, *1phhero 2phase* and *2pheroG 2phase,* The coverage measures in the other comparisons do not provide any clear preference information between the different direction changes tested but in some situations, visualization of the differences of EAFs is possible

1phero 2phase (50x10-2)	I_C	Kolmogorov-Smirnov test
$\lambda = 0$□ 1 / $\lambda = 1$□ 0	34%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0/ $\lambda = 0$□ 1	57%	
$\lambda = 0$□ 1 / $\lambda = 0$□ 1□ 0	31%	The two attainment surfaces differ somewhere
$\lambda = 0$□ 1□ 0 / $\lambda = 0$□ 1	59%	
$\lambda = 0$□ 1 / $\lambda = 1$□ 0□ 1	24%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1 / $\lambda = 0$□ 1	60%	
$\lambda = 1$□ 0 / $\lambda = 0$□ 1□ 0	41%	h_0 not rejected
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0	50%	
$\lambda = 1$□ 0 / $\lambda = 1$□ 0□ 1	39%	h_0 not rejected
$\lambda = 1$□ 0□ 1 / $\lambda = 1$□ 0	49%	
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0□ 1	32%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1/ $\lambda = 0$□ 1□ 0	55%	

Table 27: Results for *1phero* 2phase for direction changes (50x10-2)

The Figures 53, 54 and 55 suggest that a direction changes $\lambda = 0$□ 1 for *1phero 2phase* performs worse than the other types of direction changes tested. It is especially worse for points with a small makespan (in the left bottom corner) if we compare with the solutions obtained when the algorithm begins wit the makespan as most important objective (direction changes beginning with $\lambda = 1$). For the other regions, differences are very small.

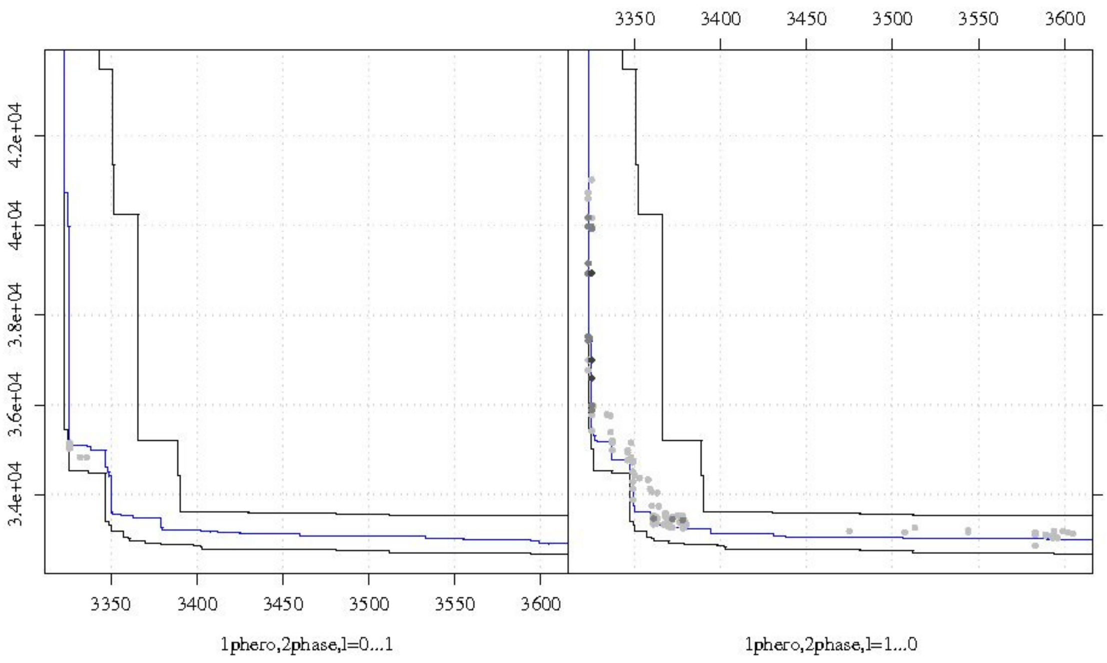


Figure 53: Differences of EAFs, *1phero 2phase* for direction changes,(50x10-2)

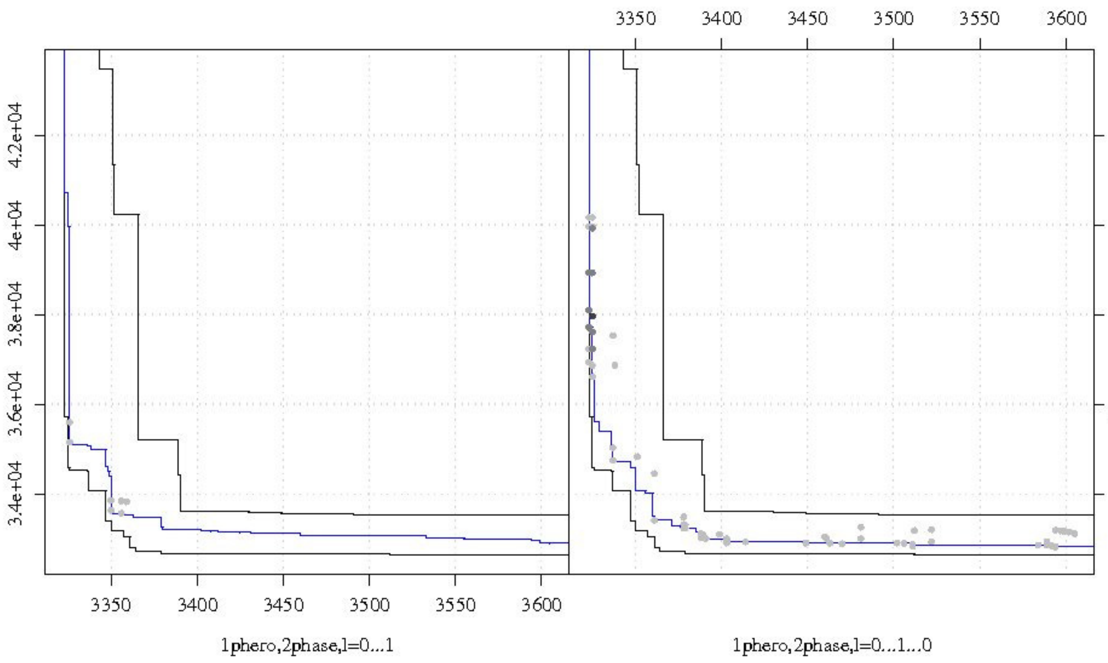


Figure 54: Differences of EAFs, *1phero 2phase* for direction changes ,(50x10-2)

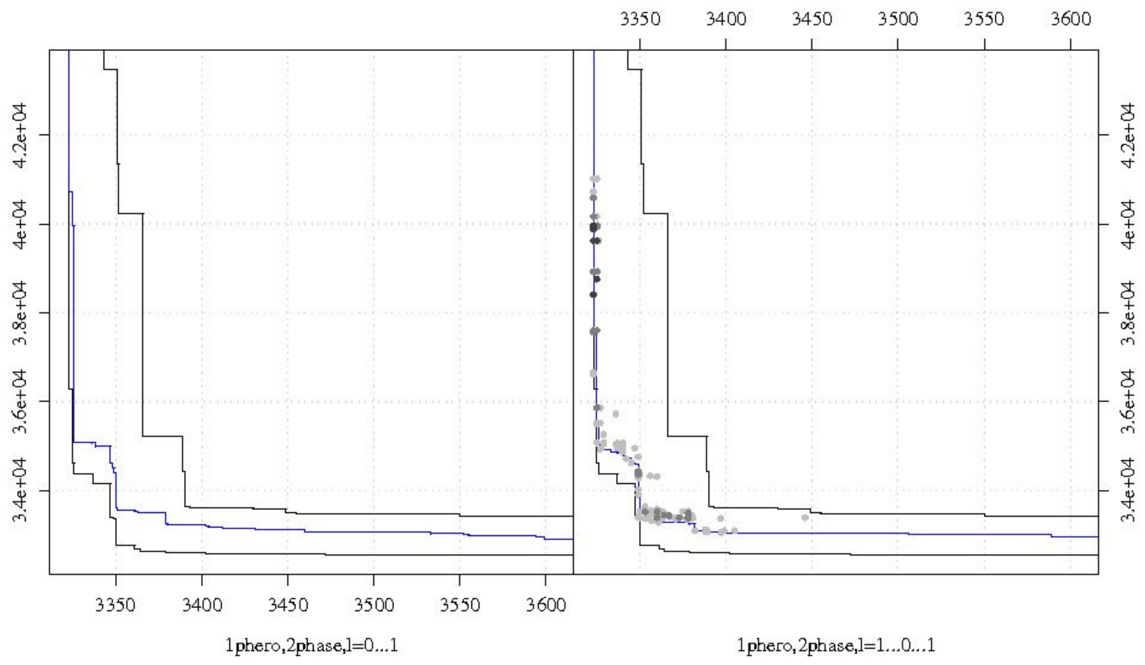


Figure 55: Differences of EAFs, *1phero 2phase* for direction changes ((50x10-2)

A comparison of the two double 2phase suggests that the use of a direction changes of type $\lambda = 1 ... 0 ... 1$ performs better in the left region tan the double 2phase $\lambda = 0 \ldots 1 \ldots 0$. This is illustrated in Figure 56.

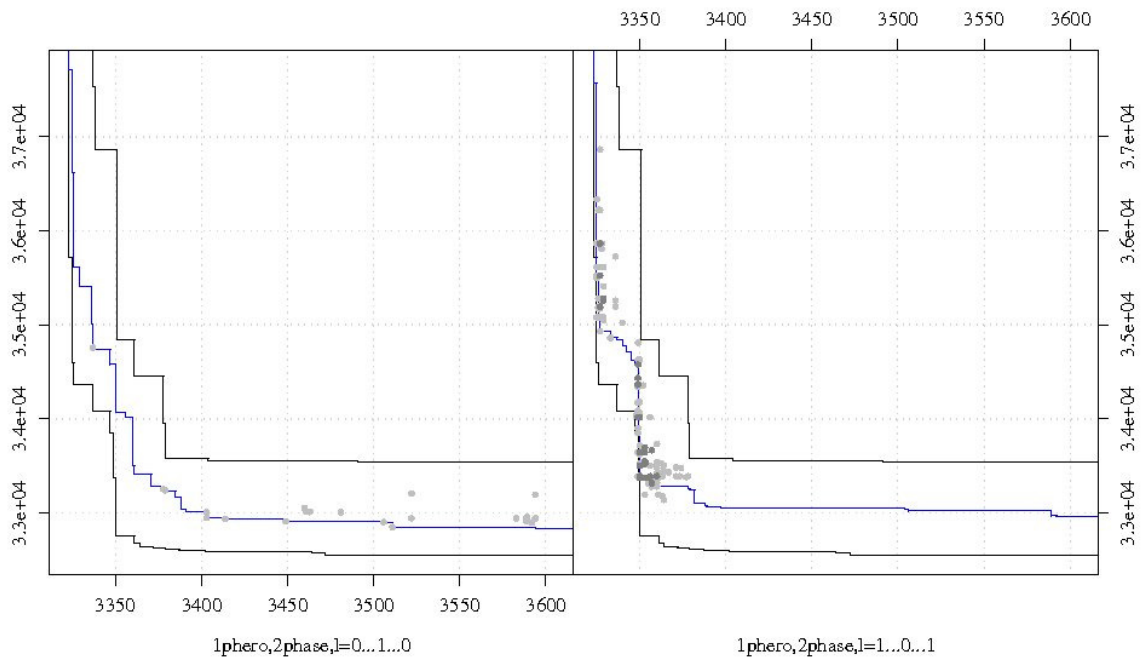


Figure 56: Differences of EAFs, *1phero* 2phase for direction changes ((50x10-2)

2pheroG 2phase (50x10-2)	I_C	Kolmogorov-Smirnov test

$\lambda = 0$□ 1 / $\lambda = 1$□ 0	43%	h_0 not rejected
$\lambda = 1$□ 0/ $\lambda = 0$□ 1	54%	
$\lambda = 0$□ 1 / $\lambda = 0$□ 1□ 0	39%	h_0 not rejected
$\lambda = 0$□ 1□ 0 / $\lambda = 0$□ 1	55%	
$\lambda = 0$□ 1 / $\lambda = 1$□ 0□ 1	61%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1 / $\lambda = 0$□ 1	33%	
$\lambda = 1$□ 0 / $\lambda = 0$□ 1□ 0	35%	h_0 not rejected
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0	57%	
$\lambda = 1$□ 0 / $\lambda = 1$□ 0□ 1	58%	h_0 not rejected
$\lambda = 1$□ 0□ 1 / $\lambda = 1$□ 0	39%	
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0□ 1	60%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1/ $\lambda = 0$□ 1□ 0	33%	

Table 28: Results for *2pheroG* 2phase for different direction changes (50x10-2)

Figures 57 and 58 show that the type beginning by $\lambda = 0$ performs better than the type $\lambda = 1$□ 0□ 1 for the middle region of the objective space.

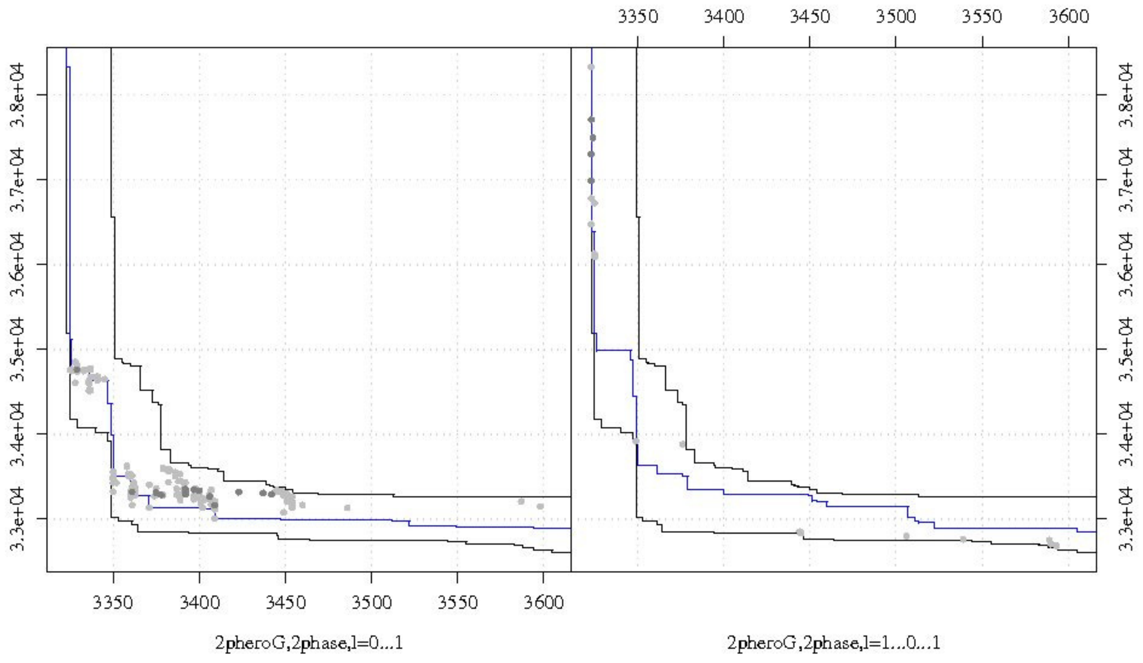


Figure 57: Differences of EAFs, *2pheroG 2phase* for direction changes (50x10-2)

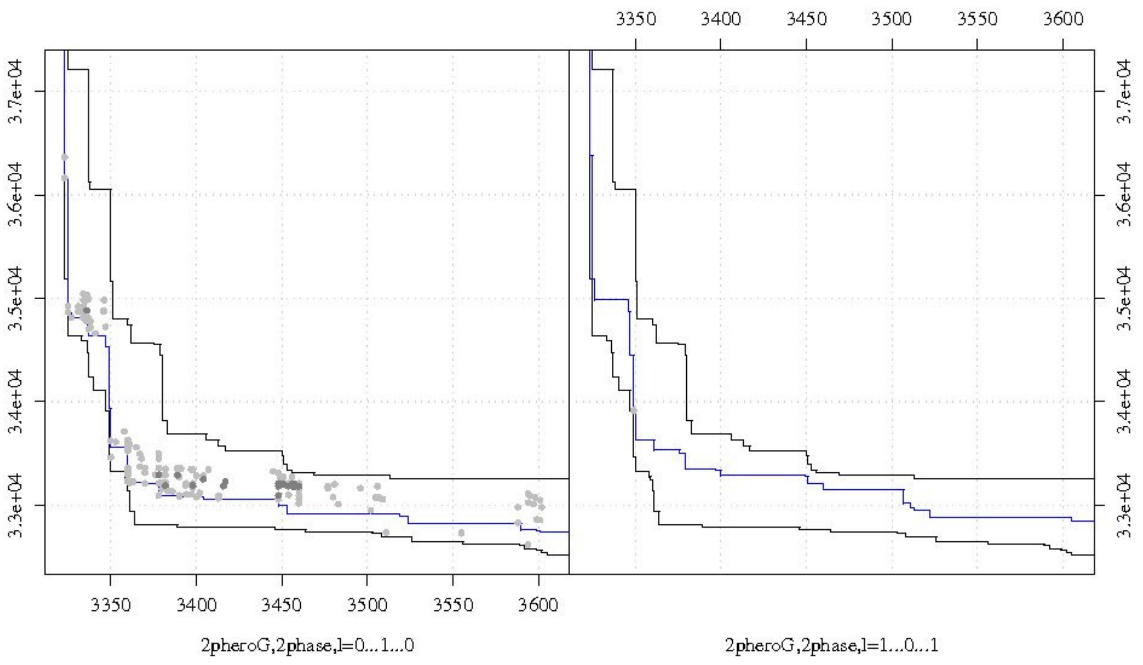


Figure 58: Differences of EAFs, *2pheroG 2phase* for direction changes (50x10-2)

2pheroL 2phase – (50x10-2)	I_C	Kolmogorov-Smirnov test
$\lambda = 0\square\ 1$ / $\lambda = 1\square\ 0$	77%	The two attainment surfaces differ somewhere
$\lambda = 1\square\ 0$/ $\lambda = 0\square\ 1$	25%	
$\lambda = 0\square\ 1$ / $\lambda = 0\square\ 1\square\ 0$	42%	h_0 not rejected
$\lambda = 0\square\ 1\square\ 0$ / $\lambda = 0\square\ 1$	53%	
$\lambda = 0\square\ 1$ / $\lambda = 1\square\ 0\square\ 1$	44%	h_0 not rejected
$\lambda = 1\square\ 0\square\ 1$ / $\lambda = 0\square\ 1$	55%	
$\lambda = 1\square\ 0$ / $\lambda = 0\square\ 1\square\ 0$	28%	The two attainment surfaces differ somewhere
$\lambda = 0\square\ 1\square\ 0$ / $\lambda = 1\square\ 0$	70%	
$\lambda = 1\square\ 0$ / $\lambda = 1\square\ 0\square\ 1$	26%	The two attainment surfaces differ somewhere
$\lambda = 1\square\ 0\square\ 1$ / $\lambda = 1\square\ 0$	74%	
$\lambda = 0\square\ 1\square\ 0$ / $\lambda = 1\square\ 0\square\ 1$	48%	h_0 not rejected
$\lambda = 1\square\ 0\square\ 1$/ $\lambda = 0\square\ 1\square\ 0$	47%	

Table 29: Results for *2pheroL* 2phase for different direction changes (50x10-2)

Table 28 and the Figure 61 suggest that the type $\lambda = 1\square\ 0$ is generally less performing than the three other direction changes. The same table and the Figures 59 and 60 suggest that a direction changes of type $\lambda = 0\square\ 1$ gives best performances for the configuration *1phero 2 phase* tested on 50x30-2.

1phero 2phase 50x30-2	I_C	Kolmogorov-Smirnov test
$\lambda = 0$□ 1 / $\lambda = 1$□ 0	82%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0/ $\lambda = 0$□ 1	11%	
$\lambda = 0$□ 1 / $\lambda = 0$□ 1□ 0	61%	The two attainment surfaces differ somewhere
$\lambda = 0$□ 1□ 0 / $\lambda = 0$□ 1	31%	
$\lambda = 0$□ 1 / $\lambda = 1$□ 0□ 1	60%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1 / $\lambda = 0$□ 1	24%	
$\lambda = 1$□ 0 / $\lambda = 0$□ 1□ 0	19%	The two attainment surfaces differ somewhere
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0	73%	
$\lambda = 1$□ 0 / $\lambda = 1$□ 0□ 1	27%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1 / $\lambda = 1$□ 0	65%	
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0□ 1	47%	h_0 not rejected
$\lambda = 1$□ 0□ 1/ $\lambda = 0$□ 1□ 0	46%	

Table 30: Results for *1phero* 2phase for different direction changes (50x30-2)

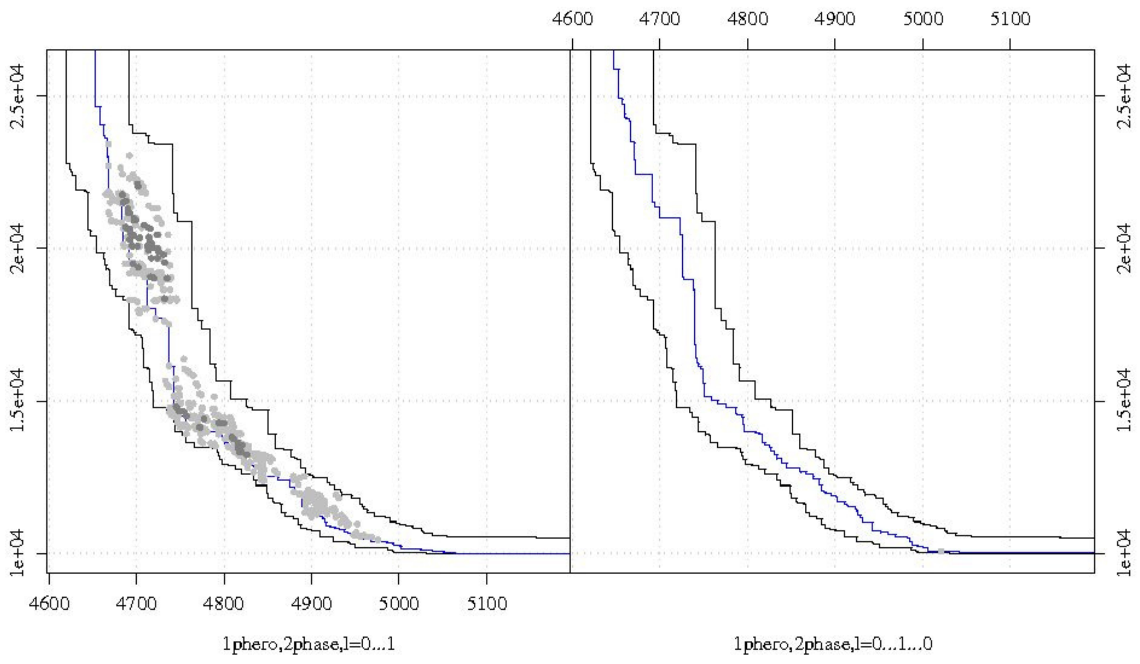


Figure 59 : Differences of EAFs, *1phero 2phase* for direction changes 1,50x30-2)

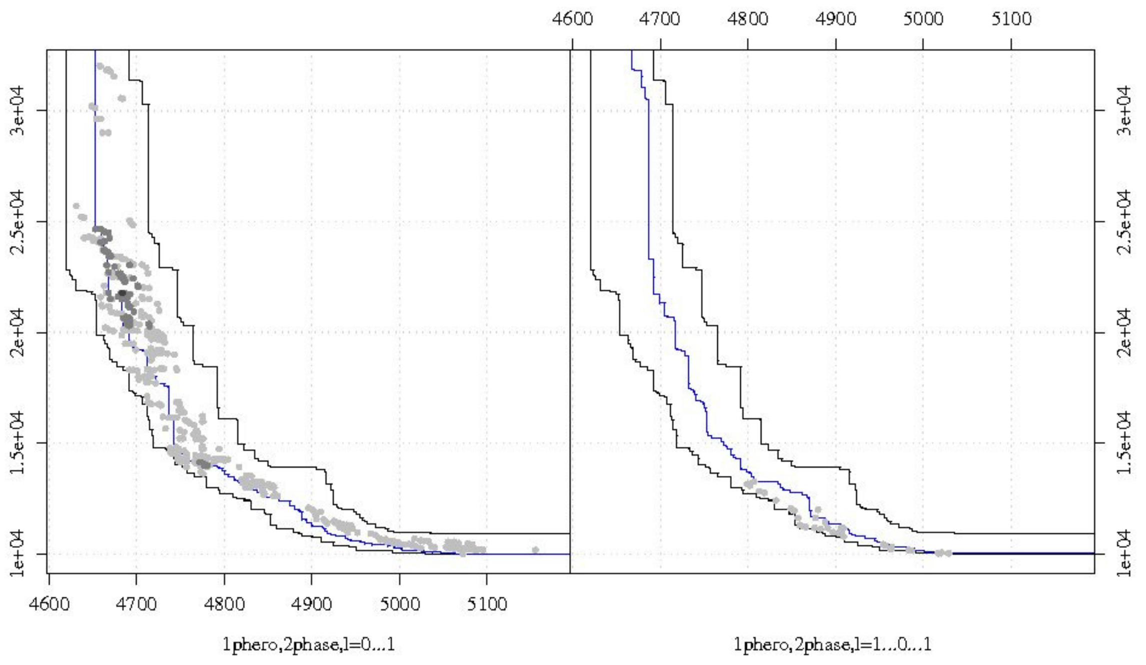


Figure 60: Differences of EAFs, *1phero 2phase* for direction changes (2,50x30-2)

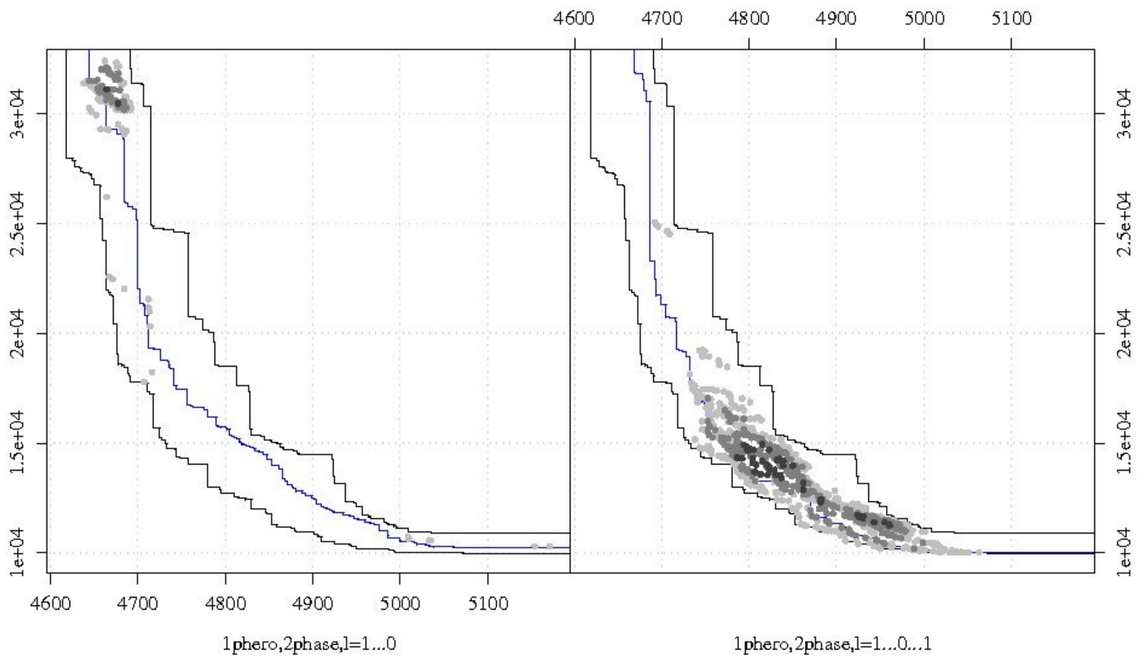


Figure 61: Differences of EAFs, *1phero 2phase* for direction changes (3,50x30-2)

In the Table §1 the results of the same comparison for *2pheroG 2phase* are presented. This table and the Figure 62indicate that direction changes beginning with $\lambda = 1$ give generally worse results than direction changes beginning with $\lambda = 0$.

2pheroG 2phase - 50x30-2	I_C	Kolmogorov-Smirnov test
$\lambda = 0$□ 1 / $\lambda = 1$□ 0	69%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0/ $\lambda = 0$□ 1	26%	
$\lambda = 0$□ 1 / $\lambda = 0$□ 1□ 0	43%	h_0 not rejected
$\lambda = 0$□ 1□ 0 / $\lambda = 0$□ 1	49%	
$\lambda = 0$□ 1 / $\lambda = 1$□ 0□ 1	75%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1 / $\lambda = 0$□ 1	19%	
$\lambda = 1$□ 0 / $\lambda = 0$□ 1□ 0	18%	The two attainment surfaces differ somewhere
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0	75%	
$\lambda = 1$□ 0 / $\lambda = 1$□ 0□ 1	40%	h_0 not rejected
$\lambda = 1$□ 0□ 1 / $\lambda = 1$□ 0	51%	
$\lambda = 0$□ 1□ 0 / $\lambda = 1$□ 0□ 1	76%	The two attainment surfaces differ somewhere
$\lambda = 1$□ 0□ 1/ $\lambda = 0$□ 1□ 0	17%	

Table 31: Results for *2pheroG* 2phase f for different direction changes (50x30-2)

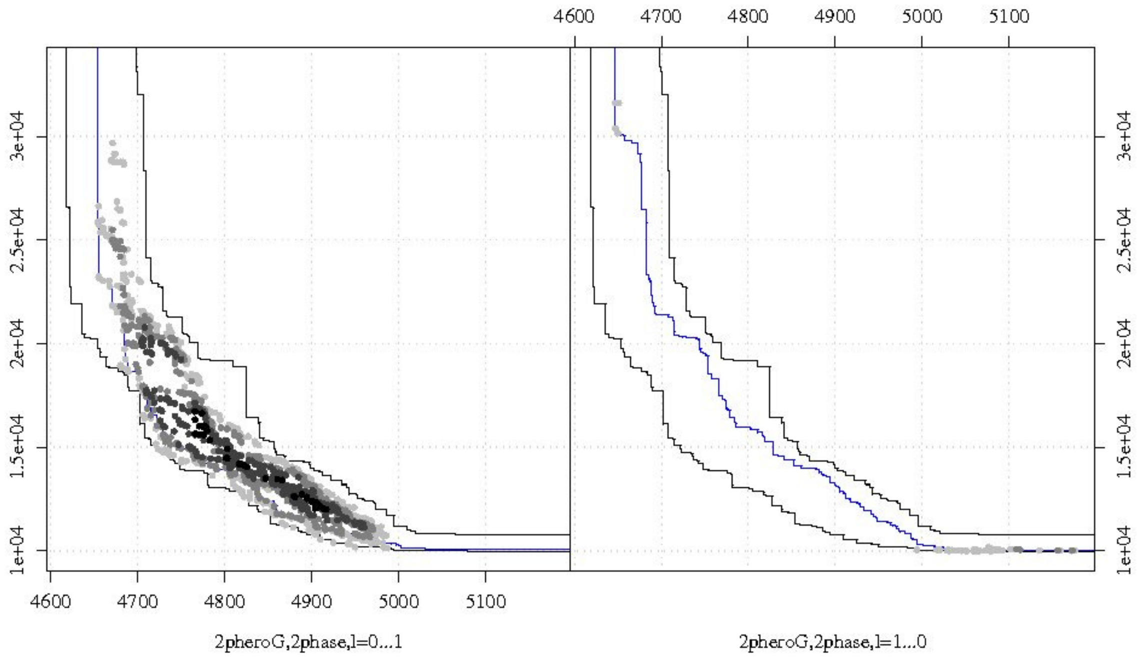


Figure 62: *2pheroG 2phase* for direction changes (50x30-2)

The Table 32 indicates that the direction changes $\lambda = 1$□ 0 is worst than the two double 2phase approaches and the Figure 62 show that it is worse than the direction changes $\lambda = 0...1$ except in the right region of the objective space. The Figure 63 show that the direction changes $\lambda = 0$□ 1 is generally worst than the double 2phase $\lambda = 1$□ 0□ 1.

2pheroL 2phase- 50x30-2	I_C	Kolmogorov-Smirnov test
$\lambda=0$□ 1 / $\lambda=1$□ 0	67%	The two attainment surfaces differ somewhere
$\lambda=1$□ 0/ $\lambda=0$□ 1	30%	
$\lambda=0$□ 1 / $\lambda=0$□ 1□ 0	41%	h_0 not rejected
$\lambda=0$□ 1□ 0 / $\lambda=0$□ 1	53%	
$\lambda=0$□ 1 / $\lambda=1$□ 0□ 1	35%	The two attainment surfaces differ somewhere
$\lambda=1$□ 0□ 1 / $\lambda=0$□ 1	60%	
$\lambda=1$□ 0 / $\lambda=0$□ 1□ 0	23%	The two attainment surfaces differ somewhere
$\lambda=0$□ 1□ 0 / $\lambda=1$□ 0	71%	
$\lambda=1$□ 0 / $\lambda=1$□ 0□ 1	24%	The two attainment surfaces differ somewhere
$\lambda=1$□ 0□ 1 / $\lambda=1$□ 0	72%	
$\lambda=0$□ 1□ 0 / $\lambda=1$□ 0□ 1	43%	h_0 not rejected
$\lambda=1$□ 0□ 1/ $\lambda=0$□ 1□ 0	43%	

Table 32: Results for *2pheroL* for different direction changes (50x30-2)

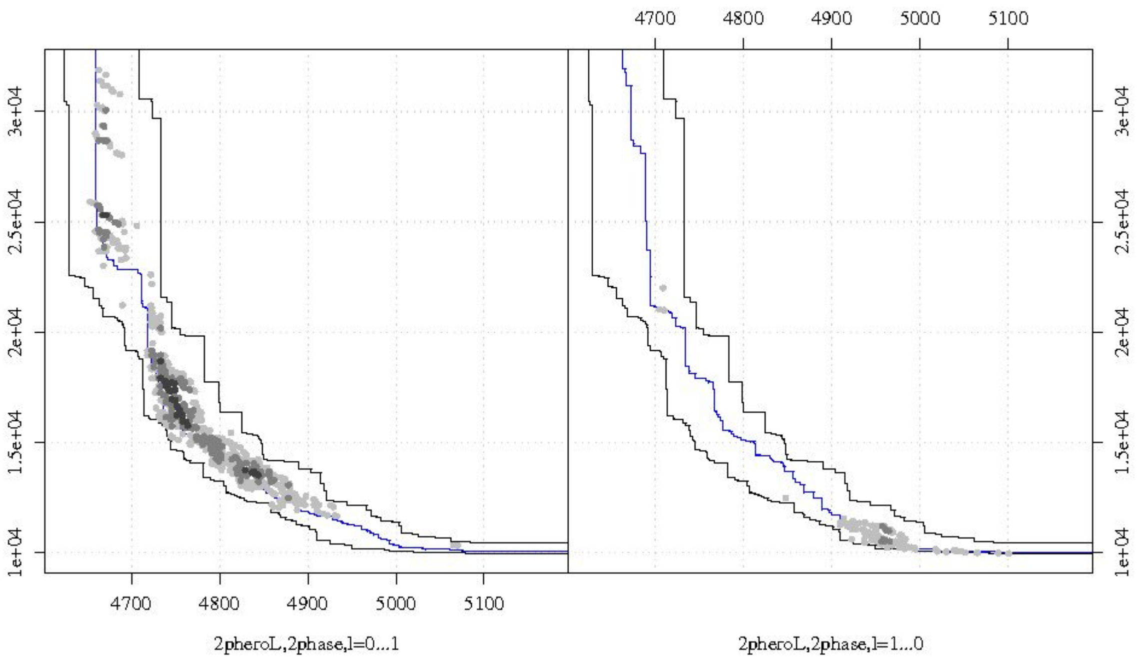


Figure 63: *2pheroL 2phase* for direction changes (1,50x30-2)

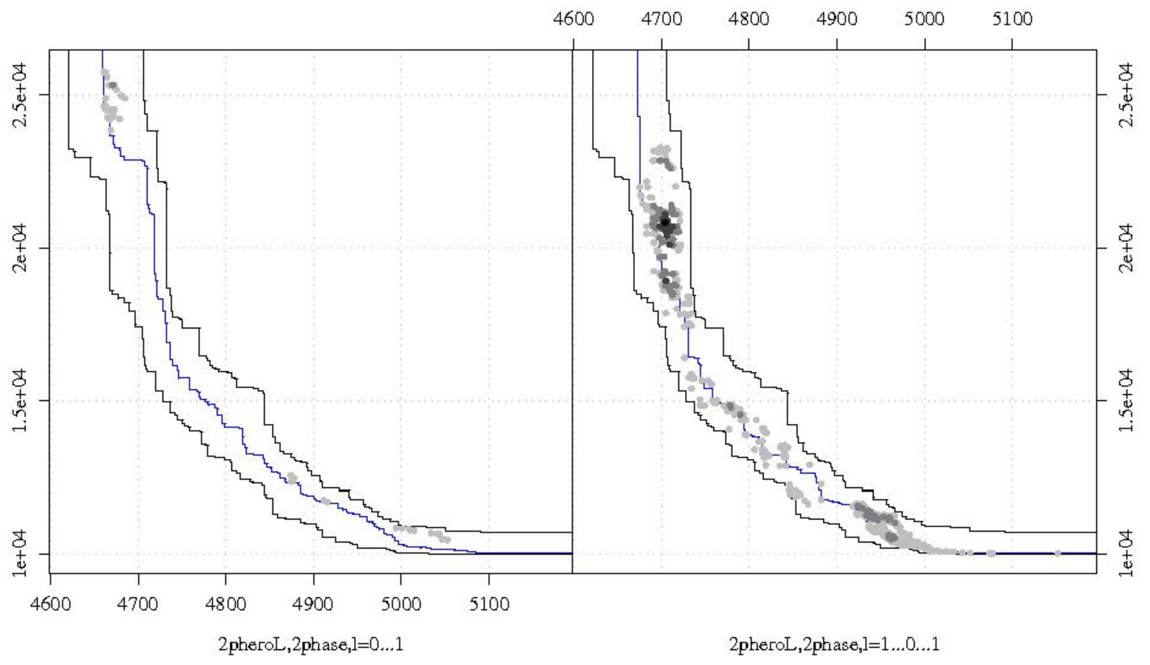


Figure 64: ***2pheroL 2phase*** **for direction changes (2, 50x30-2)**

Summary of the observations of the influence of different direction changes

For 50x10-2 (50 jobs, 10 machines):

- For *1phero 2phase*, it seems preferable to use a direction changes beginning with $\lambda = 1$ if the preferred objective is the makespan. The double 2phase approach $\lambda = 1 \ldots 0 \ldots 1$ is slightly better than the direction changes $\lambda = 1 \ldots 0$, even if the differences are not large. If the decision maker looks for solutions in the middle and in the right region of the objective space, a direction changes of type $\lambda = 0 \ldots 1 \ldots 0$ could be more appropriated

- For *2pheroG 2phase*, the direction changes $\lambda = 1 \ldots 0 \ldots 1$ seems to be less performing than direction changes beginning with $\lambda = 0$. Between these two types, the double 2phase approach seems to be slightly better if we refer to the hypervolume indicator

- For 2pheroL *2phase*, the direction changes $\lambda = 1 \ldots 0$ is worse than the three other types tested and if we compare the two double 2phase approach $\lambda = 0...1...0$ and $\lambda = 1 \ldots 0 \ldots 1$, we have an indication that the second type could give better results for small values of makespan whereas the other could give slightly better solutions for the middle and the right region of the objective space.

for the 50 jobs 30 machines instance, we observe that:

- For *1phero phase,* the direction changes $\lambda = 0 \to 1$ seems to be more appropriated for this instance and for most of regions of the objective space
- For *2pheroG 2phase,* a direction changes beginning with $\lambda = 0$ are clearly better for the instance and for almost each region of the objective space. Among the two, the evolution $\lambda = 0 \to 1$ seems to be slightly better
- For *2pheroL 2phase*, this time the type of direction changes $\lambda = 0 \to 1$ even if it is better than the type $\lambda = 1 \to 0$, it often performs worse than the type of direction changes $\lambda = 1 \to 0 \to 1$.

6 Influence of the non dominated local search

Here we present the details of comparisons between three configurations using the non dominated local search and the three same configurations without non dominated local search. The three configurations *1phero 2phase*, *2pheroG 2phase* and *2pheroL 2phase* have been tested on the second 50x10 instance and the second 50x30 machines instance for $|W| = 41$ and $\lambda = 1$□ 0. Tables

1phero 2phase - **λ = 1□ 0** - (50x10-2)	I_C	Kolmogorov-Smirnov test
(*1phero 2phase* with ND_LS-*1phero 2phase* without ND_LS	59%	The two attainment surfaces differ somewhere
(*1phero 2phase* without ND_LS-*1phero 2phase* with ND_LS	25%	
(2pheroG 2phase with ND_LS-1phero2phase without ND_LS	53%	The two attainment surfaces differ somewhere
(2pheroG 2phase without ND_LS-2pheroG 2phase with ND_LS	30%	
(2pheroL 2phase with ND_LS-2pheroL 2phase without ND_LS	44%	h_0 not rejected
(2pheroL 2phase without ND_LS-2pheroL 2phase with ND_LS	50%	

Table 33: Results with/without ND_LS (50x10-2)

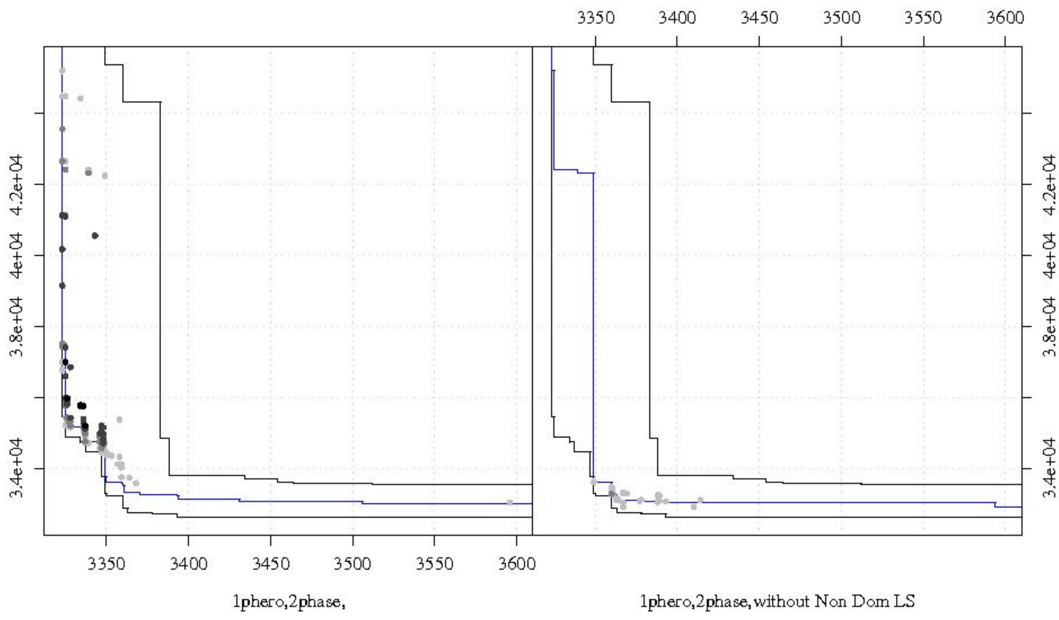


Figure 65: Differences of EAFs,*1phero 2phase* with/without ND_LS (50x10-2)

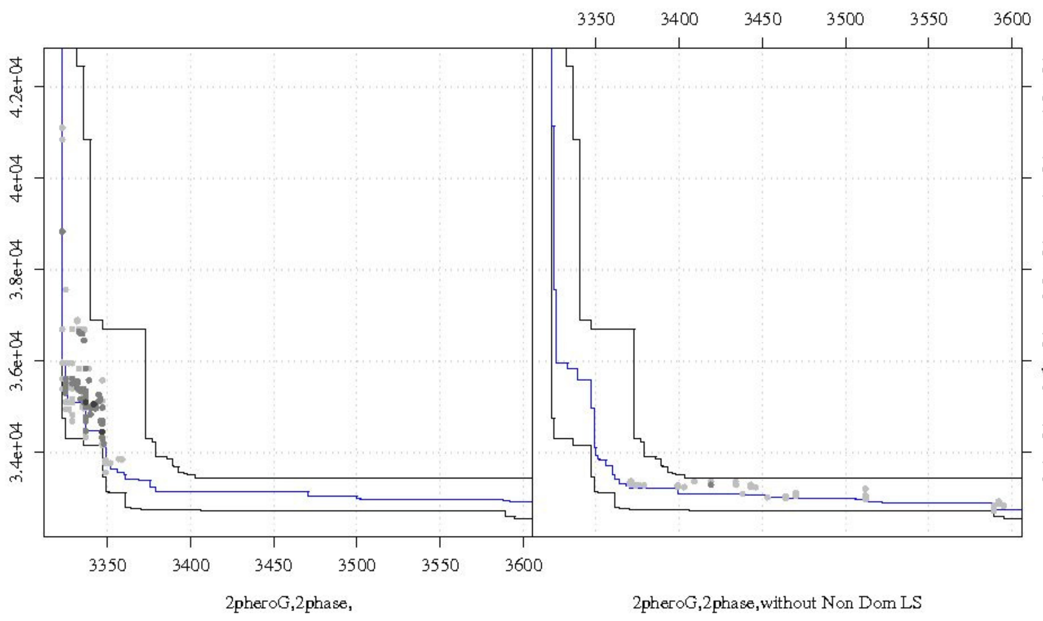


Figure 66: Differences of EAFs,2pheroG 2phase with/without ND_LS (50x10-2)

1phero 2phase - **λ = 1□ 0** - (50x30-2)	I_C	Kolmogorov-Smirnov test
(*1phero 2phase* with ND_LS-*1phero 2phase* without ND_LS	45%	h_0 not rejected
(*1phero 2phase* without ND_LS-*1phero 2phase* with ND_LS	20%	
(*2pheroG 2phase* with ND_LS-*1phero2phase* without ND_LS	40%	h_0 not rejected
(*2pheroG 2phase* without ND_LS-*2pheroG 2phase* with ND_LS	47%	
(*2pheroL 2phase* with ND_LS-*2pheroL 2phase* without ND_LS	72%	The two attainment surfaces differ somewhere
(2pheroL 2phase without ND_LS-*2pheroL 2phase* with ND_LS	23%	

Table 34 : Results with/without ND_LS (50x30-2)

7 Comparison *1phero – 2phero* approach

For this comparison, we have analyzed the differences when using *1phero* and *2phero* approach on the four different instances. We have first use a direction changes of type $\lambda = 1...0$ and then we have tested the influence of other direction changes on the result of the comparison between *1phero* and *2phero*. The Table 35 presents the results for the tests on 50x10-1. We can observe that the coverage measures suggest that *1phero scratch* is dominated by *2phero scratch*.

$\lambda = 1 \searrow 0$ - (50x10-1)	I_C	Kolmogorov-Smirnov test
(1pheroscratch/2pheroG scratch)	13%	The two attainment surfaces differ somewhere
(2pheroG scratch/1phero scratch)	85%	
(1phero 2phase/2pheroG 2phase)	39	The two attainment surfaces differ somewhere
(2pheroG 2phase/1phero 2phase)	50	
(1pheroscratch/2pheroL scratch)	13%	The two attainment surfaces differ somewhere
(2pheroL scratch/1phero scratch)	83%	
(1phero 2phase/2pheroL 2phase)	43%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	46%	

Table 35: Results of a comparison *1phero/2phero* (50x10-1)

The Figures 67 and 68 suggest that 2phero 2phase help slightly to have better results for non dominated solutions with small makespan.

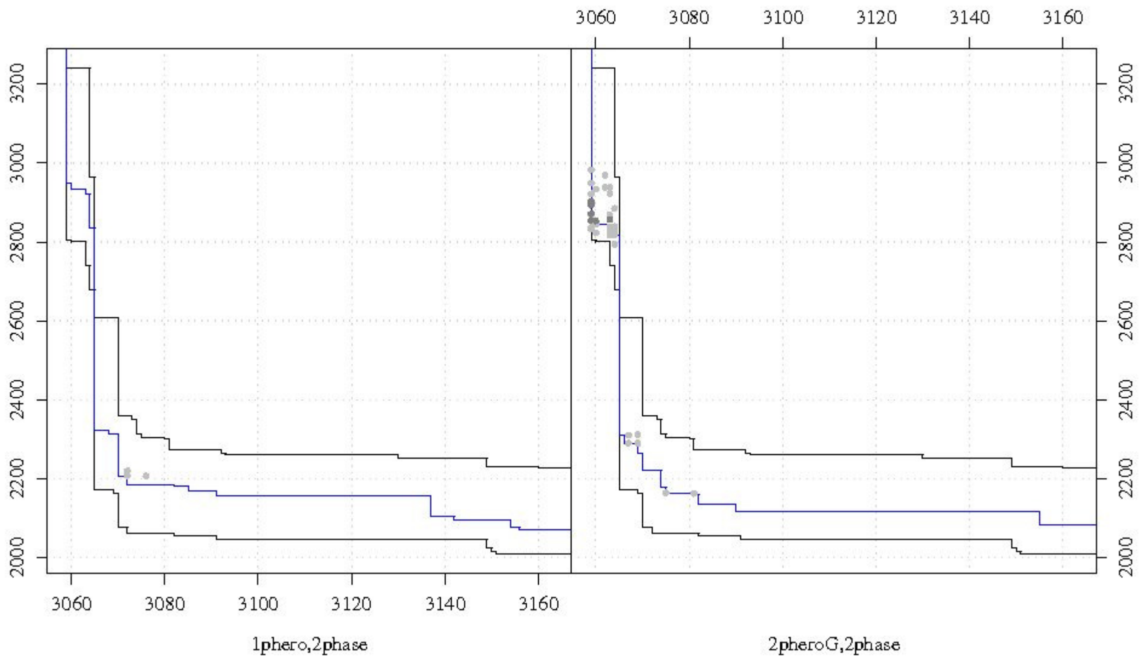


Figure 67: Differences of EAFs, *1phero 2phase/2pheroG 2phase* (50x10-1)

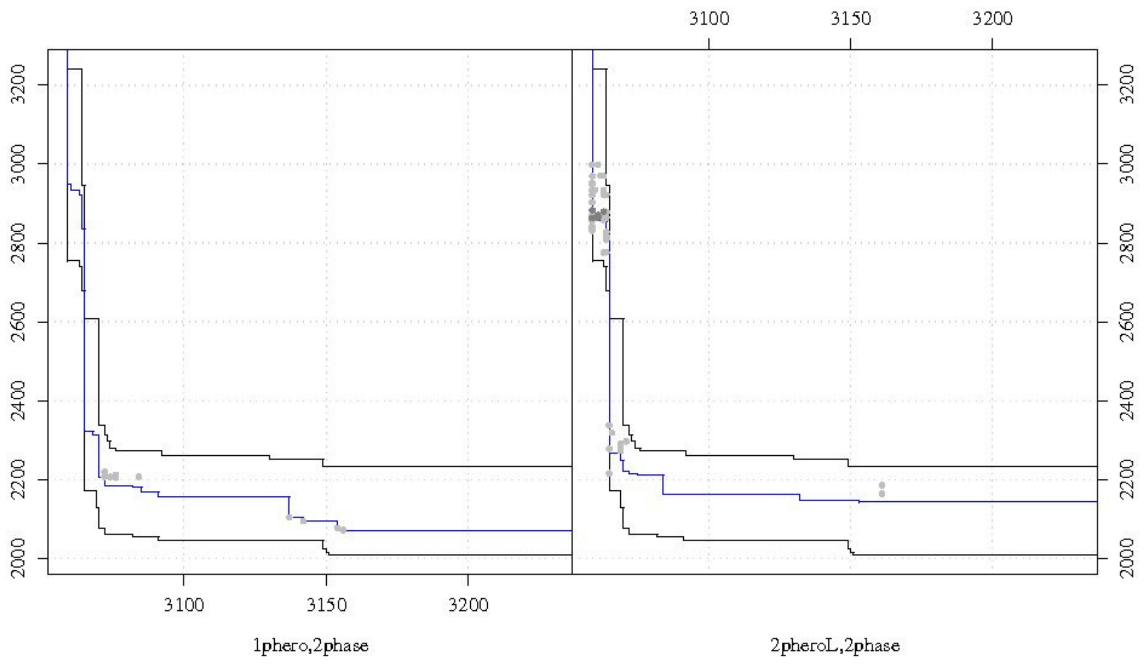


Figure 68: Differences of EAFs, *1phero 2phase*/*2pheroL 2phase* (50x10-1)

The Table 36 presents results obtained for 50x10-2. This time, no clear preference is provided by the coverage measure and the EAFs are significantly different only for one comparison. The Figure 68 illustrates this difference and indicates that *2pheroL 2phase* seems to be more performing in finding good solutions with small makespan, but is slightly worse for other regions of the objective space.

λ = 1 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(1pheroscratch/2pheroG scratch)	39%	h_0 not rejected
(2pheroG scratch/1phero scratch)	48%	
(1phero 2phase/2pheroG 2phase)	42	h_0 not rejected
(2pheroG 2phase/1phero 2phase)	51	
(1pheroscratch/2pheroL scratch)	59%	h_0 not rejected
(2pheroL scratch/1phero scratch)	36%	
(1phero 2phase/2pheroL 2phase)	51%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	46%	

Table 36: Results of a comparison *1phero/2phero* (50x10-2)

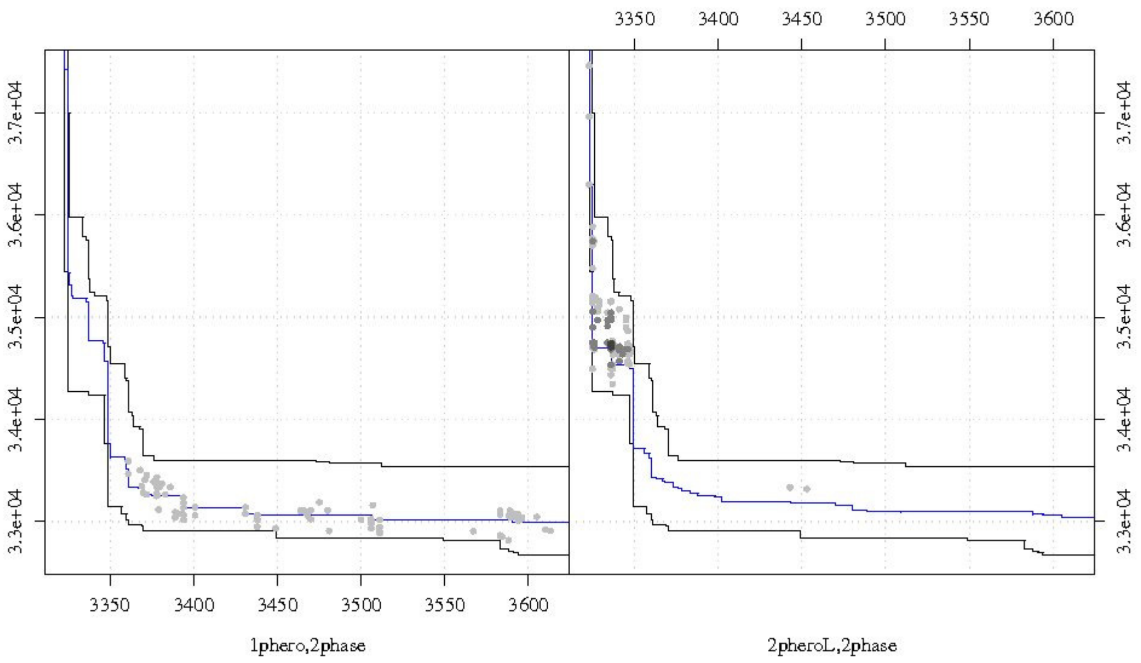


Figure 69: Differences of EAFs, *1phero 2phase/2pheroL 2phase* (50x10-2)

Only one outperformance relation can be observed in the Table 33 for the tests on the 3rdinstance, *2pheroL 2phase* is better than *1phero 2phase*. But the analysis of the difference s of EAFs is possible. Figure69 and 70 show that *1phero 2* performs better in the middle while *2pheroG 2phase* could be preferred if one of the extreme region of the objective space.

The Figure 71 suggests the possibility of dividing the search in two operations, one operation using *1phero scratch* for the left region and the use of *2pheroL scratch* for the right region.

λ = 1□ 0 - 50x30-1	I_C	Kolmogorov-Smirnov test
(1pheroscratch/2pheroG scratch)	54%	The two attainment surfaces differ somewhere
(2pheroG scratch/1phero scratch)	38%	
(1phero 2phase/2pheroG 2phase)	49%	The two attainment surfaces differ somewhere
(2pheroG 2phase/1phero 2phase)	43%	
(1pheroscratch/2pheroL scratch)	51%	The two attainment surfaces differ somewhere
(2pheroL scratch/1phero scratch)	43%	
(1phero 2phase/2pheroL 2phase)	23%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	71%	

Table 37: Results of a comparison *1phero/2phero* (50x30-1)

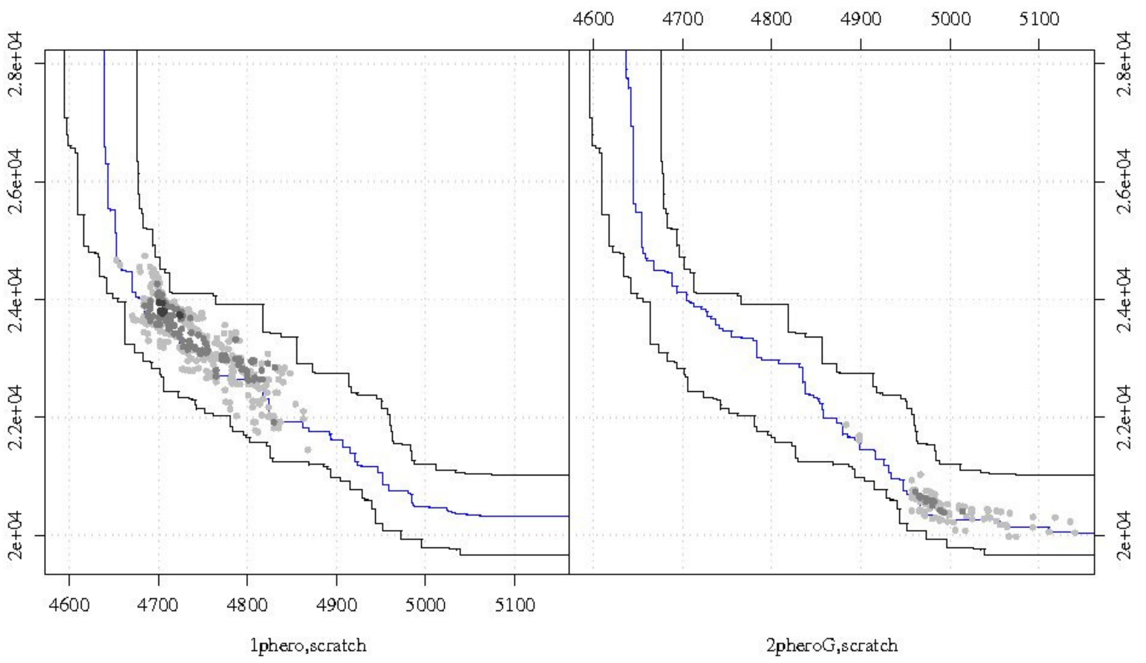


Figure 70: Differences of EAFs, *1phero scratch/2pheroG scratch* (50x30-1)

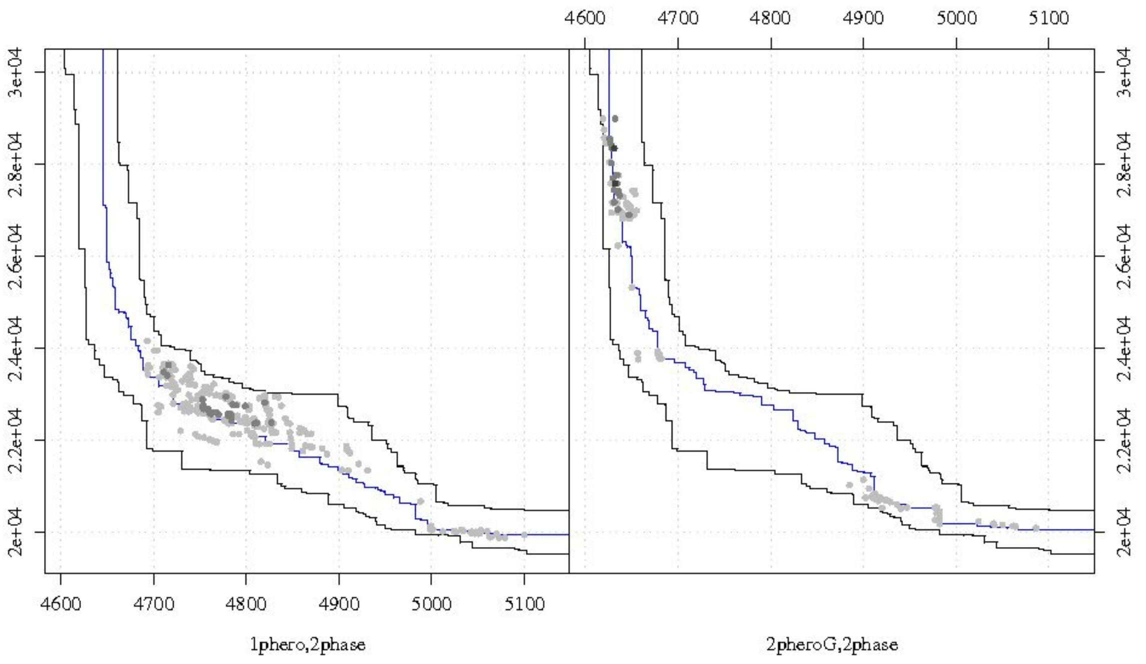


Figure 71: Differences of EAFs, *1phero 2phase/2pheroG 2phase* (50x30-1)

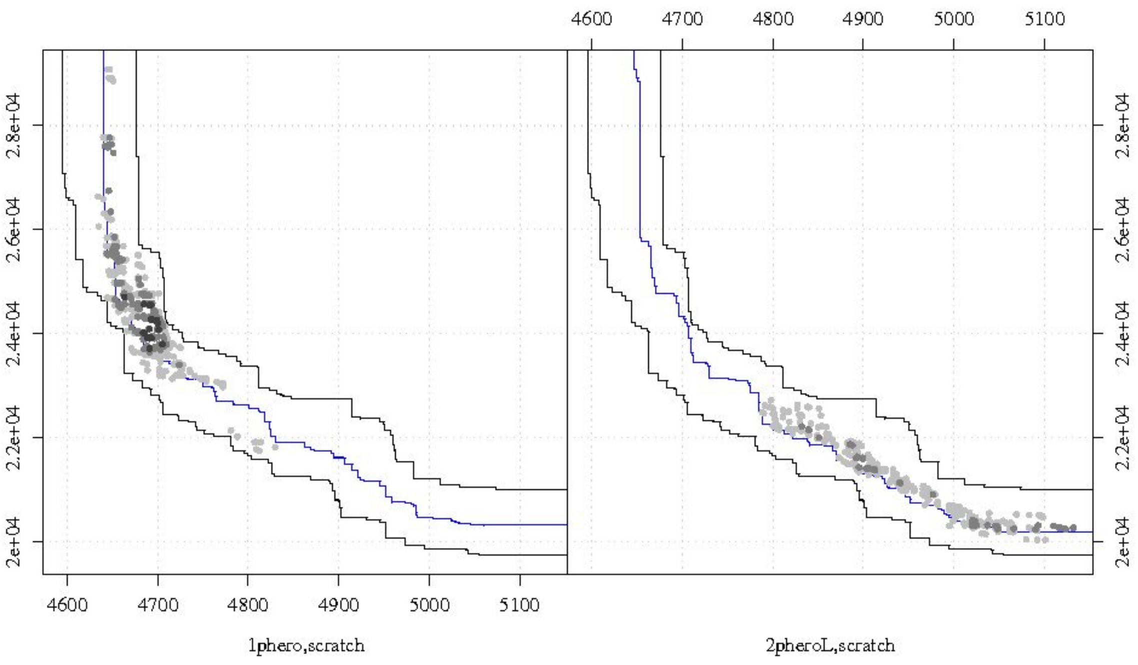


Figure 72: Differences of EAFs, 1phero *scratch/2pheroL scratch* (50x30-1)

The Table 38 presents the results for the comparison *1phero-2phero* for 50x30-2 where no clear relation of preference can be observed. The Figures 71, 72 and 73 suggest that for 50x30-2, *1phero* is not capable of finding good solutions in the right region of the objective space

λ = 1□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
(1pheroscratch/2pheroG scratch)	46%	The two attainment surfaces differ somewhere
(2pheroG scratch/1phero scratch)	49%	
(1phero 2phase/2pheroG 2phase)	51%	The two attainment surfaces differ somewhere
(2pheroG 2phase/1phero 2phase)	39%	
(1pheroscratch/2pheroL scratch)	51%	h_0 not rejected
(2pheroL scratch/1phero scratch)	40%	
(1phero 2phase/2pheroL 2phase)	31%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	62%	

Table 38: Results of a comparison *1phero/2phero* (50x30-2)

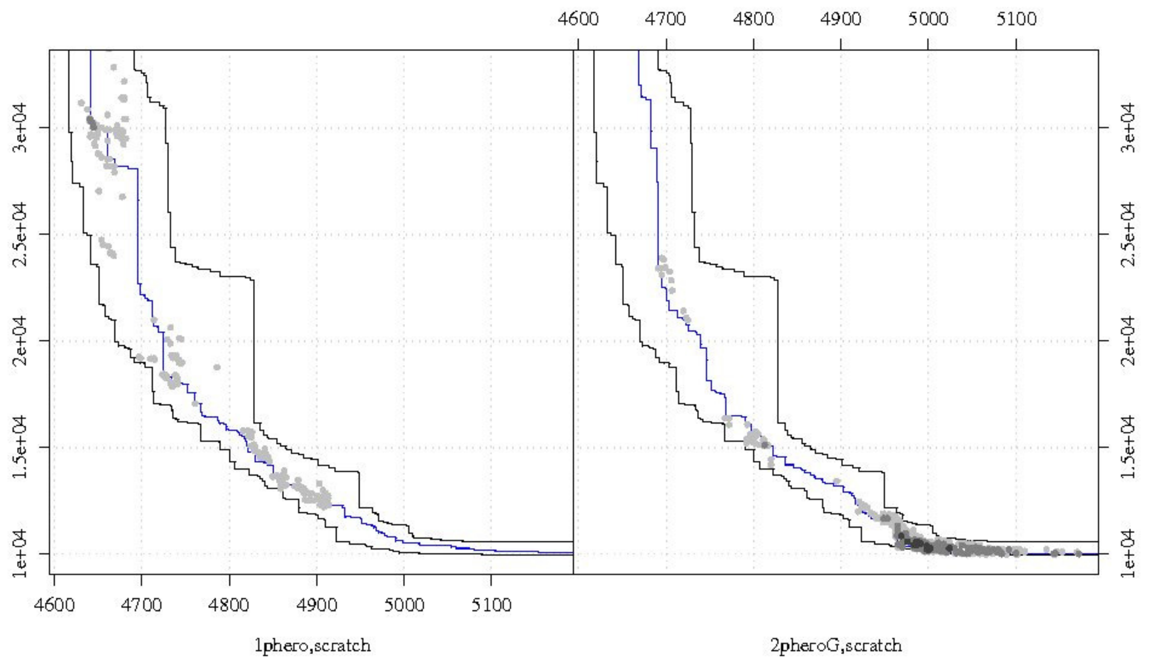


Figure 73: Differences of EAFs, *1phero scratch/2pheroG scratch* (50x30-2)

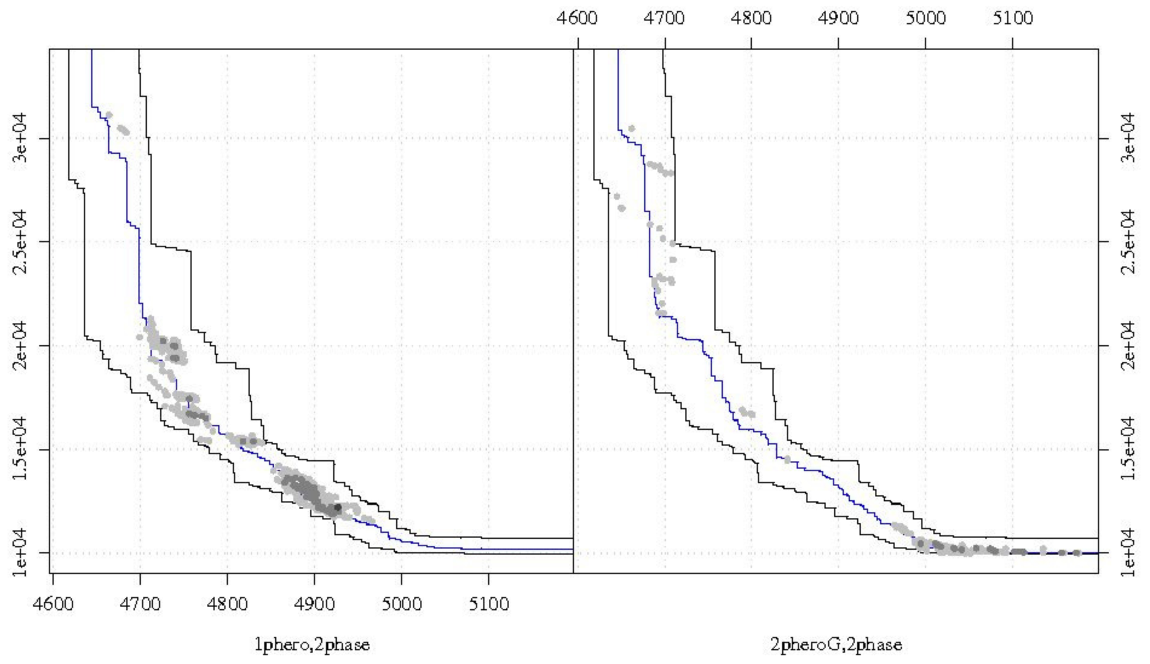


Figure 74: Differences of EAFs, *1phero 2phase/2pheroG 2phase* (50x30-2)

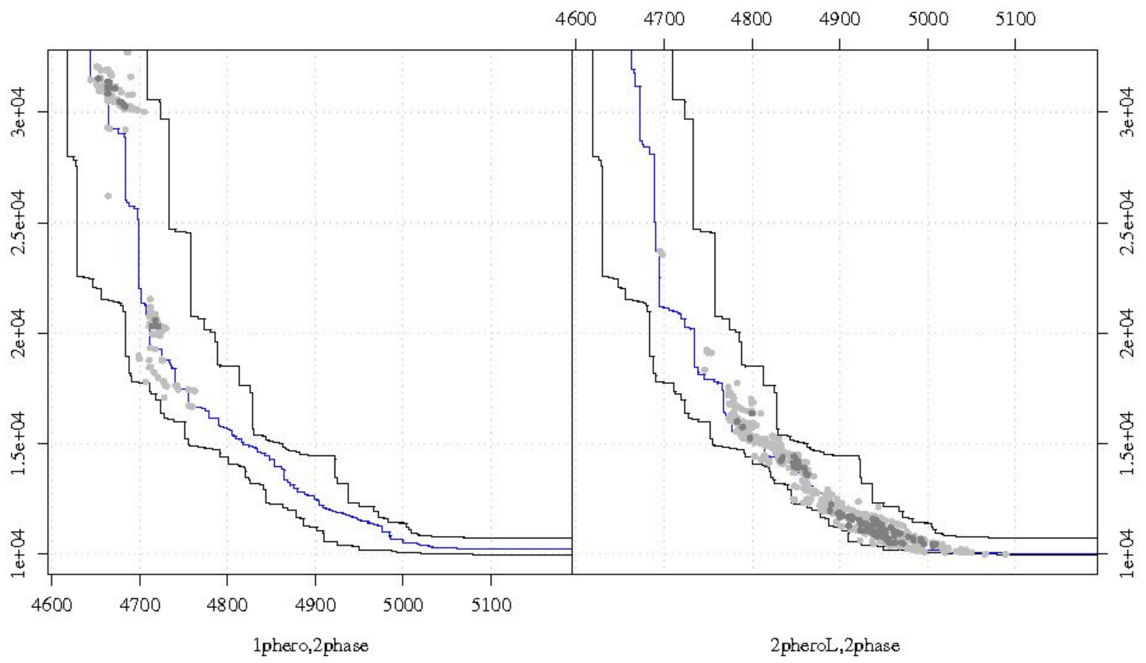


Figure 75: Differences of EAFs, *1phero 2phase/2pheroL 2phase* (50x30-2)

It must be noticed that the results obtained with other direction changes can be slightly different we will now present the results obtained when using the three different direction changes:

- $\lambda = 0$□ 1
- $\lambda = 0$□ 1□ 0
- $\lambda = 1$□ 0□ 1

with the configurations (*1phero 2phase*, *2pheroG 2phase* and *2pherooL 2phase*) on the second and 50x30-2.

The Tables 39, 40 and 41 give the results of the comparison for the three other direction changes. They do not provide any velar preference, but generally the use of the plot of differences of two EAFs is possible for the tests on (50x10-2).

$\lambda = 0$□ 1 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(1phero 2phase/2pheroG 2phase)	30%	The two attainment surfaces differ somewhere
(2pheroG 2phase/1phero 2phase)	62%	
(1phero 2phase/2pheroL 2phase)	30%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	63%	

Table 39: Results of a comparison *1phero/2phero* for λ=0-1 (50x10-2)

Figure73 suggests tat *2pheroG 2phase* is preferable to *1phero 2phase* while no clear differences appears when the direction changes was $\lambda = 1\square\ 0$.

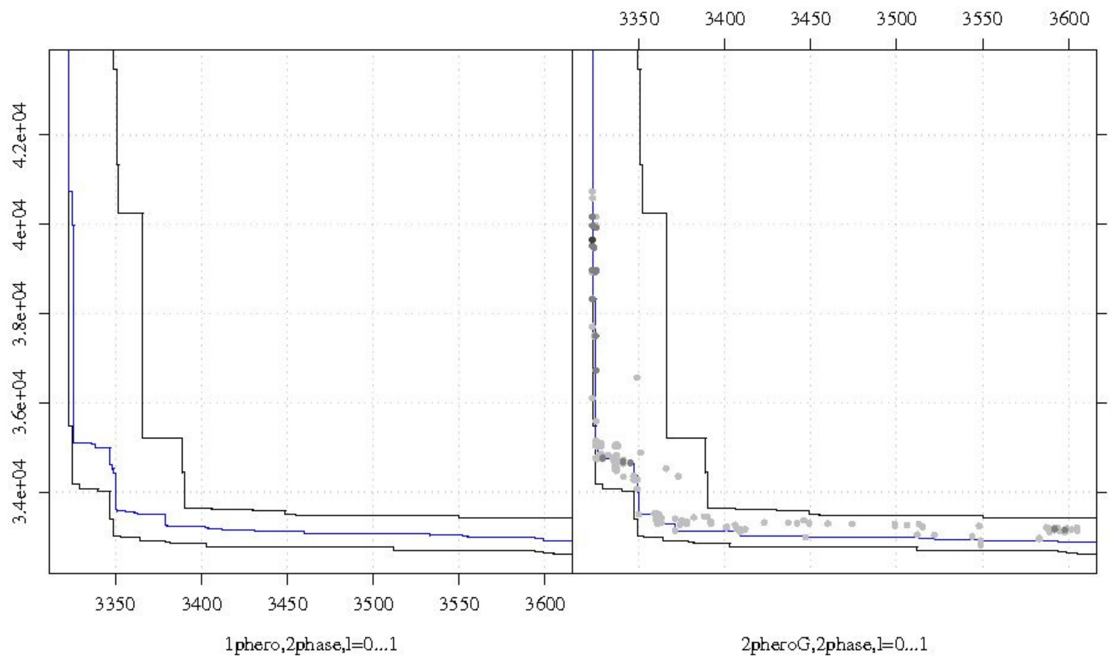


Figure 76: Differences of EAFs, *1phero 2phase/2pheroG 2phase for* λ=0-1 (50x10-2)

By contrast with the comparison for the direction changes $\lambda = 1\square\ 0$, the configuration *2pheroL 2phase* seems slightly better than *1phero 2phase* for most regions of the objective space and not only for a region with a small makespan.

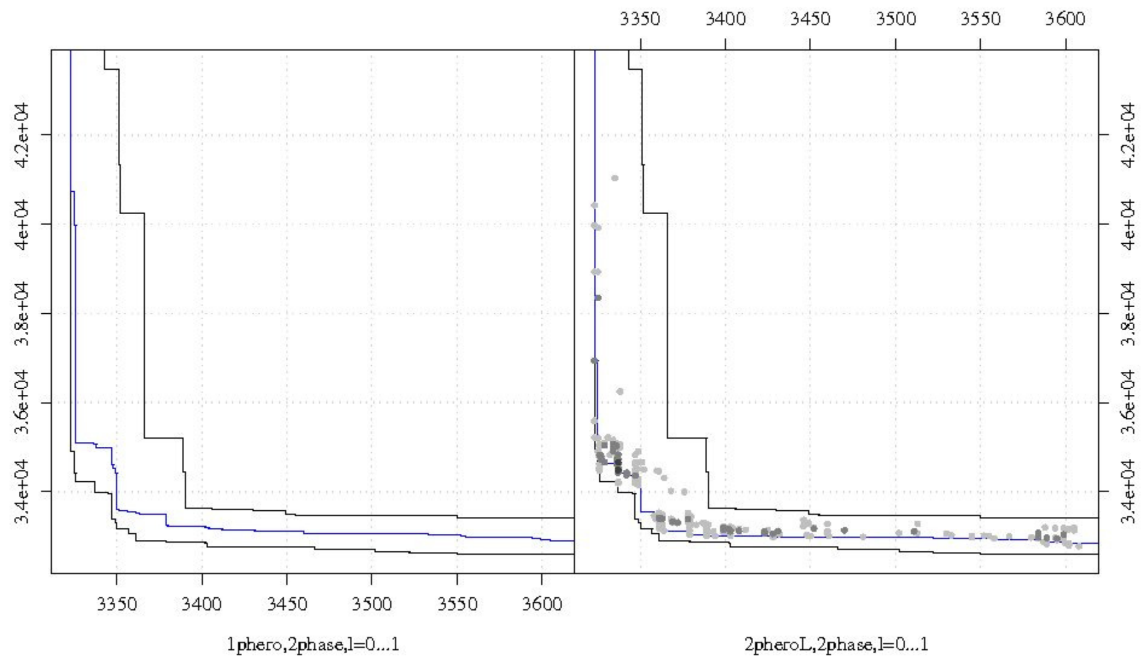


Figure 77: Differences of EAFs, *1phero 2phase/2pheroL 2phase for* λ=0-1 (50x10-2)

$\lambda = 0$□ 1□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(1phero 2phase/2pheroG 2phase)	33%	The two attainment surfaces differ somewhere
(2pheroG 2phase/1phero 2phase)	55%	
(1phero 2phase/2pheroL 2phase)	30%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	61%	

Table 40: Results of a comparison *1phero/2phero* for λ=0-1-0 (50x10-2)

The Figures 76 and 79 and 80 show that *2pheroG 2phase* and *2pheroL 2phase* have the same kind of behaviour when they are compared to *1phero 2phase.* They show that 2phero approach gives slightly better results with a double 2phase direction changes, especially for the left region of the objective space. With a direction changes of type $\lambda = 1$□ 0, the 2phero approaches were better for the right part of the front.

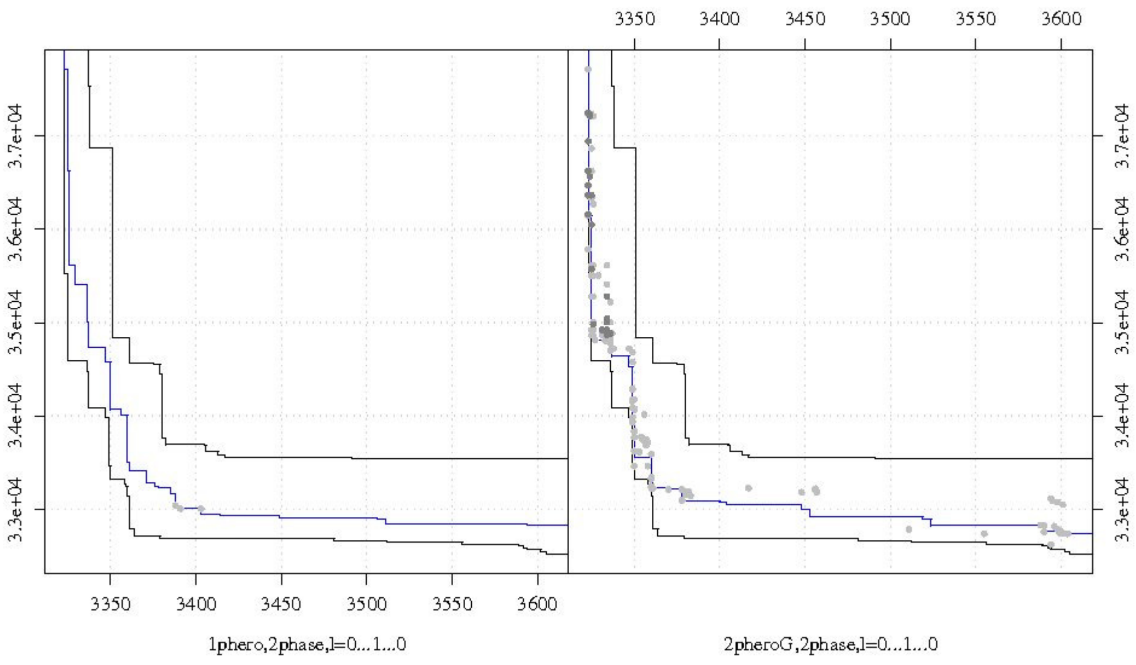


Figure 78: Differences of EAFs, *1phero 2phase/2pheroG 2phase* for λ=0-1-0 (50x10-2)

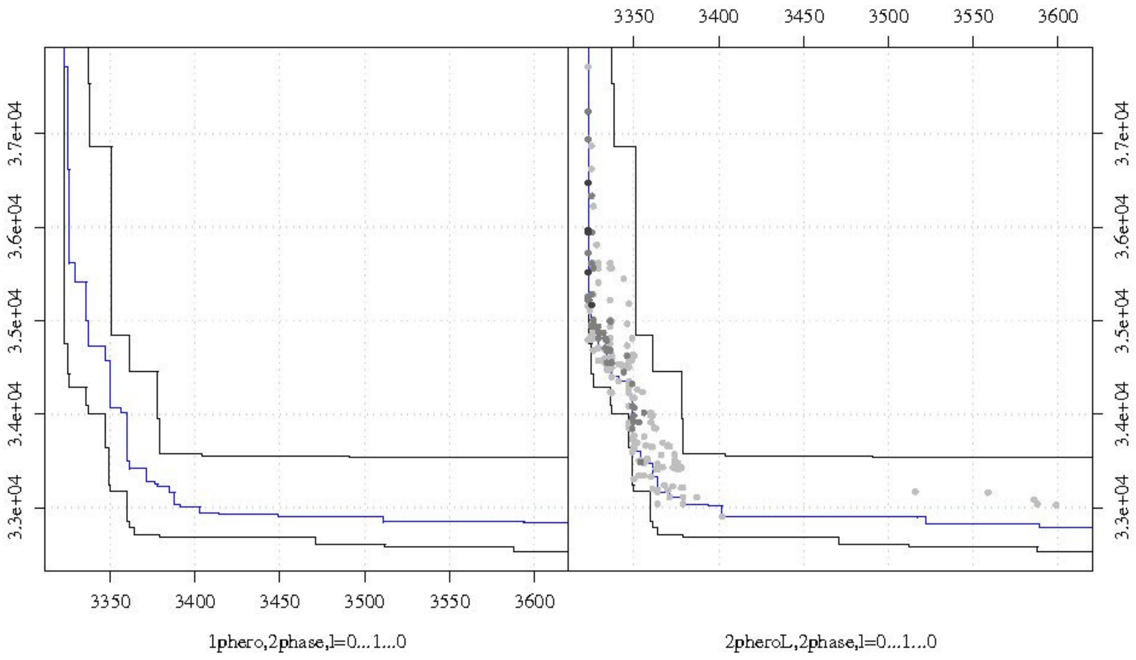


Figure 79: Differences of EAFs, *1phero 2phase/2pheroL 2phase* for λ=0-1-0 (50x10-2)

λ = 1 0 1 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(1phero 2phase/2pheroG 2phase)	542%	h_0 not rejected
(2pheroG 2phase/1phero 2phase)	34%	
(1phero 2phase/2pheroL 2phase)	33%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	57%	

Table 41: Results of a comparison *1phero/2phero* for λ=1-0-1 (50x10-2)

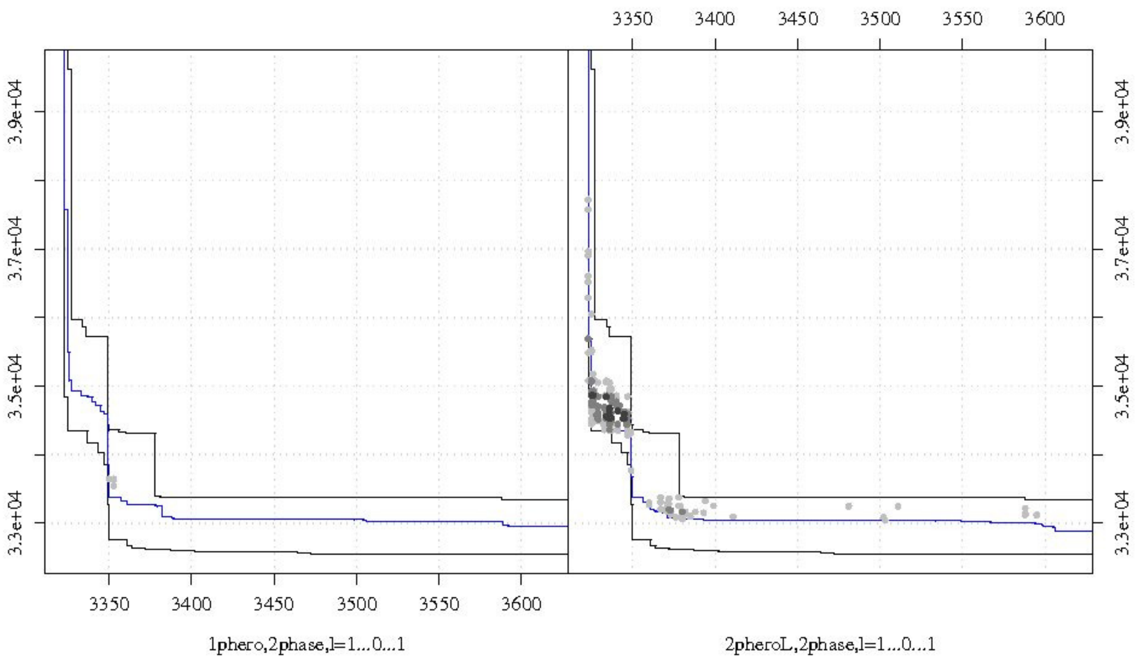


Figure 80: Differences of EAFs, *1phero 2phase/2pheroL 2phase* for λ=1-0-1 (50x10-2)

The results of the tests for 50x30-2 are presented in the Tables 42, 43 and 44. the situations where the plot of the differences of EAFs is possible are illustrated in the Figures 81 and 82. In Figure 80, the preference for *1phero 2phase* appears and not only for the extreme left upper corner. Here *1phero 2phase* seems to be better than *2pheroL 2phase* for most regions of the objective space. In the Figure 82, we observe that with the double 2phase approach, *1phero 2phase* is better than *2pheroG 2phase* and there are almost no positive differences for *2pheroG 2phase* in the extreme regions of the Pareto front anymore if compared wit the results for a direction changes $\lambda = 1 \rightarrow 0$.

λ = 0→ 1 - 50x30-2	I_C	Kolmogorov-Smirnov test
(1phero 2phase/2pheroG 2phase)	57%	h_0 not rejected
(2pheroG 2phase/1phero 2phase)	34%	
(1phero 2phase/2pheroL 2phase)	67%	The two attainment surfaces differ somewhere
(2pheroL 2phase/1phero 2phase)	23%	

Table 42: Results of a comparison *1phero/2phero* for λ=0-1 (50x30-2)

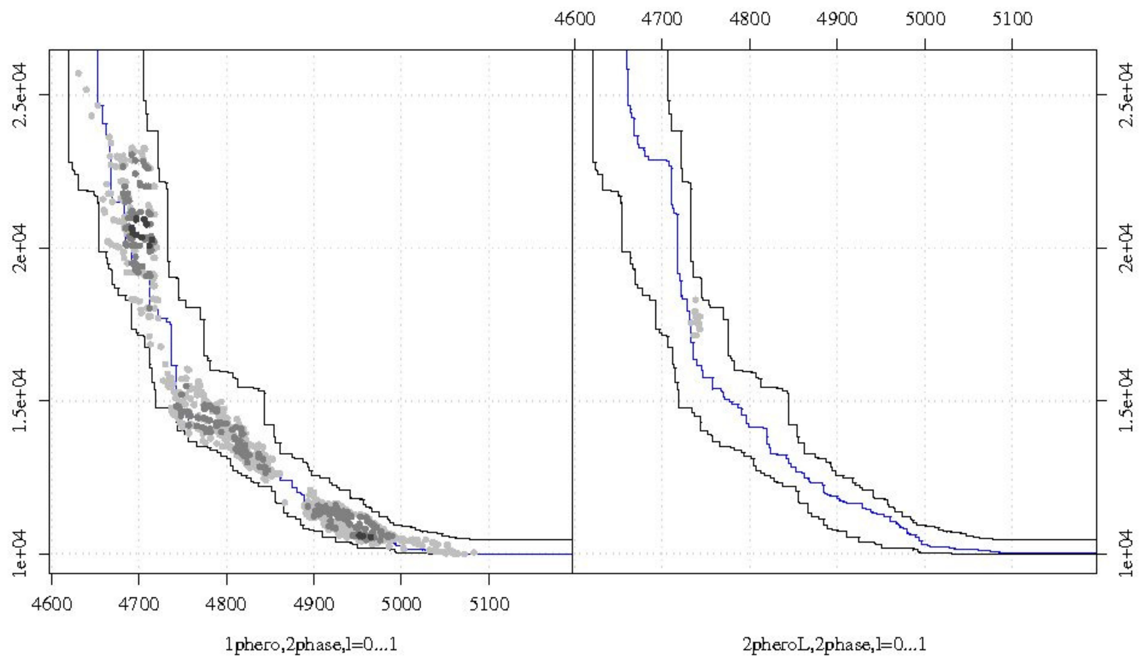


Figure 81: Differences of EAFs, *1phero 2phase /2pheroL 2phase* for λ=0-1 (50x30-2)

λ = 0□ 1□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
(1phero 2phase/2pheroG 2phase)	42%	h_0 not rejected
(2pheroG 2phase/1phero 2phase)	51%	
(1phero 2phase/2pheroL 2phase)	46%	h_0 not rejected
(2pheroL 2phase/1phero 2phase)	48%	

Table 43: Results of a comparison *1phero/2phero* for λ=0-1-0 (50x30-2)

λ = 1□ 0□ 1 - 50x30-2	I_C	Kolmogorov-Smirnov test
(1phero 2phase/2pheroG 2phase)	68%	The two attainment surfaces differ somewhere
(2pheroG 2phase/1phero 2phase)	22%	
(1phero 2phase/2pheroL 2phase)	47%	h_0 not rejected
(2pheroL 2phase/1phero 2phase)	47%	

Table 44: Results of a comparison *1phero/2phero* for λ=1-01 (50x30-2)

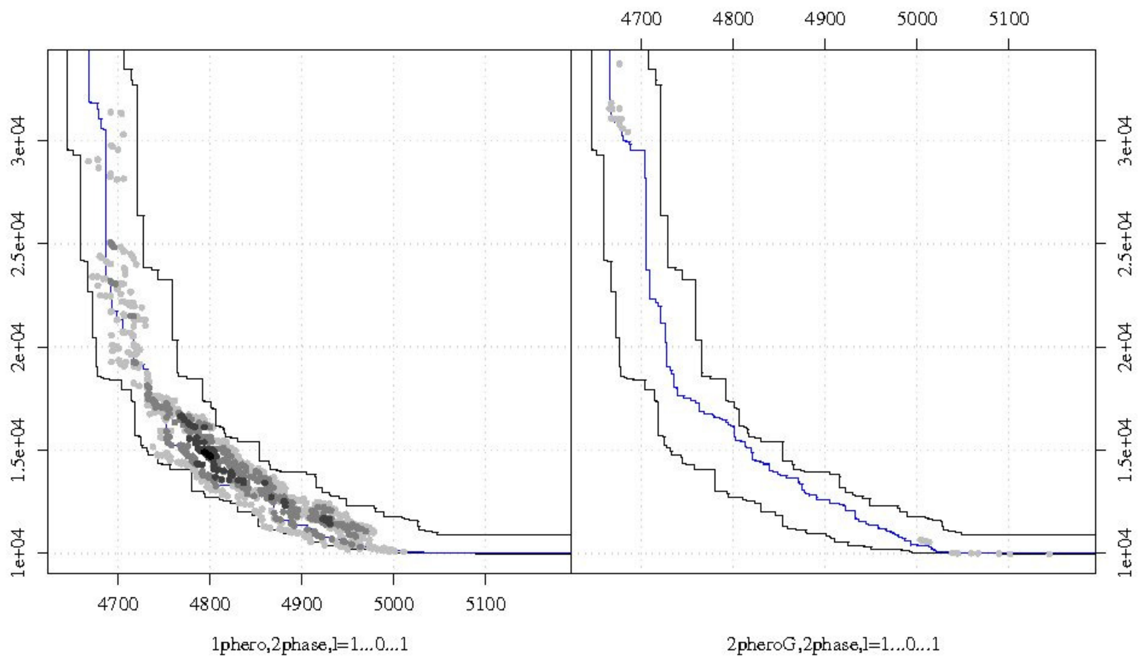


Figure 82: Differences of EAFs, *1phero 2phase /2pheroG 2phase* for λ=1-0-1 (50x30-2)

Summary of the observations on 1phero or 2*phero* approach

For the two instances 50 jobs. 10 machines, we can observe that:

- *2phero scratch* configuration is never worst than *1phero scratch* and that it often performs better
- *2pheroL 2phase and 2pheroG 2phase* configuration are preferable to *1phero 2phase* when the decision maker looks for solution with best makespan
- A direction changes beginning wit $\lambda = 0$ seems to be slightly more favourable for *2phero* approaches than for *1phero* approaches.

For the two instances 50 jobs and 30 machines, the results differ and we can observe that:

- *1phero (scratch or 2phase)* seems to be better than *2pheroG* (*scratch* or *2phas*e) in the middle whereas *2pheroG* seems be better for solutions located in the right bottom corner
- When *1phero 2phase* is compared with *2pheroL 2phase*, we observe that *2pheroL* 2phase seems to perform better than *1phero 2phase* in the middle region of the objective space for a direction changes $\lambda = 1\square\ 0$. When the direction changes chosen is $\lambda = 0...1$, the observation may be different, *1pero 2phase* seems to be better for most of the regions of the objective space for 50x30-2.

8 Comparison *scratch – 2phase* approach

Here we present the results for a comparison *scratch-2phase* for the three configurations, *1phero*, *2pheroG* and *2pheroL*. The different comparisons will be presented instance by instance. The results of these tests are presented in the Tables 45, 46, 47, 48. The situations of clear preferences have already been presented in section 5.4.6, we will not repeat them.

λ = 1□ 0 - (50x10-1)	I_C	Kolmogorov-Smirnov test
(1phero scratch/1phero 2phase)	16%	The two attainment surfaces differ somewhere
(1phero 2phase/1phero scratch)	76%	
(2pheroG scratch/2pheroG 2phase)	41%	h_0 not rejected
(2pheroG 2phase/2pheroG scratch)	49%	
(2pheroL scratch/2pheroL 2phase)	36%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroL scratch)	53%	

Table 45: Results of a comparison *scratch/2phase* (50x10-1)

In the Figure 80, we can observe that the two configurations seem to perform better than the other in two clear distinct regions. *2peroL 2phase* being better for solutions with a small makespan, *2pheroL* scratch for a region with a higher value of makespan.

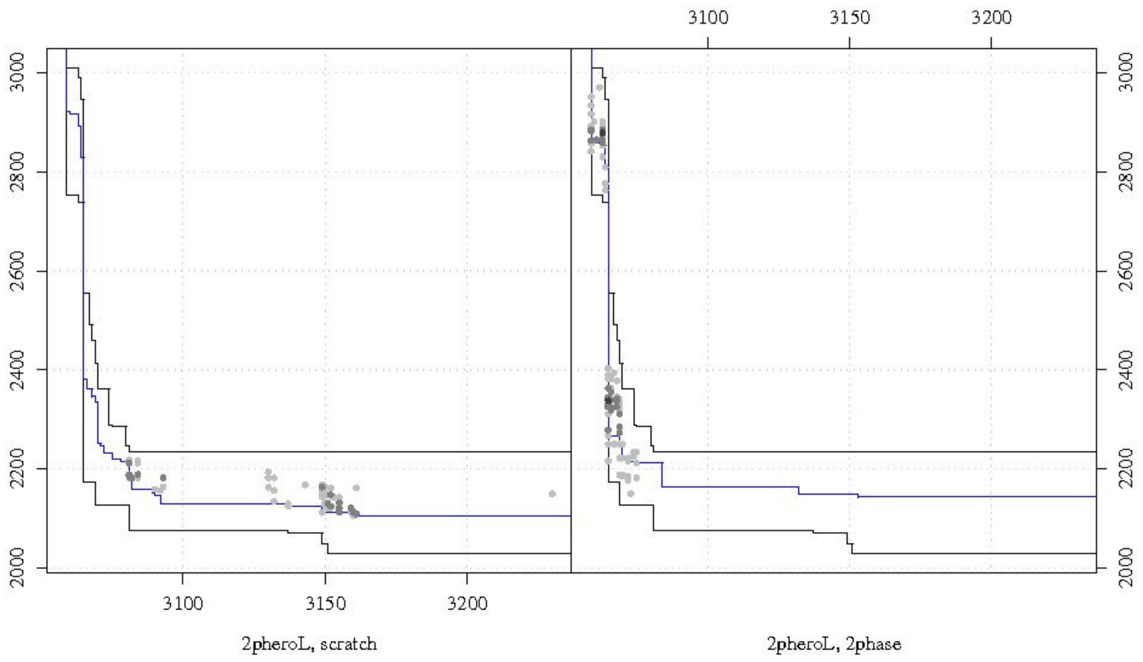


Figure 83: Differences of EAFs, *2pheroL scratch/2pheroL 2phase* (50x10-1)

λ = 1□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(1pheroscratch/1phero 2phase)	39%	h_0 not rejected
(1phero 2phase/1phero scratch)	51%	
(2pheroG scratch/2pheroG 2phase)	42%	h_0 not rejected
(2pheroG 2phase/2pheroG scratch)	48%	
(2pheroL scratch/2pheroL 2phase)	12%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroL scratch)	85%	

Table 46: Results of a comparison *scratch/2phase* (50x10-2)

λ = 1□ 0 - 50x30-1	I_C	Kolmogorov-Smirnov test
(1pheroscratch/1phero 2phase)	30%	The two attainment surfaces differ somewhere
(1phero 2phase/1phero scratch)	60%	
(2pheroG scratch/2pheroG 2phase)	35%	The two attainment surfaces differ somewhere
(2pheroG 2phase/2pheroG scratch)	58%	
(2pheroL scratch/2pheroL 2phase)	17%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroL scratch)	79%	

Table 47: Results of a comparison scratch/*2phase* (50x30-1)

Figures 83, 84and 85 provide the same observation; most of time, a decision maker will prefer to use *2phase approach* which gives better results in the middle and in the right region of the objective space and which is only slightly worse to the scratch approach for the extreme left upper corner.

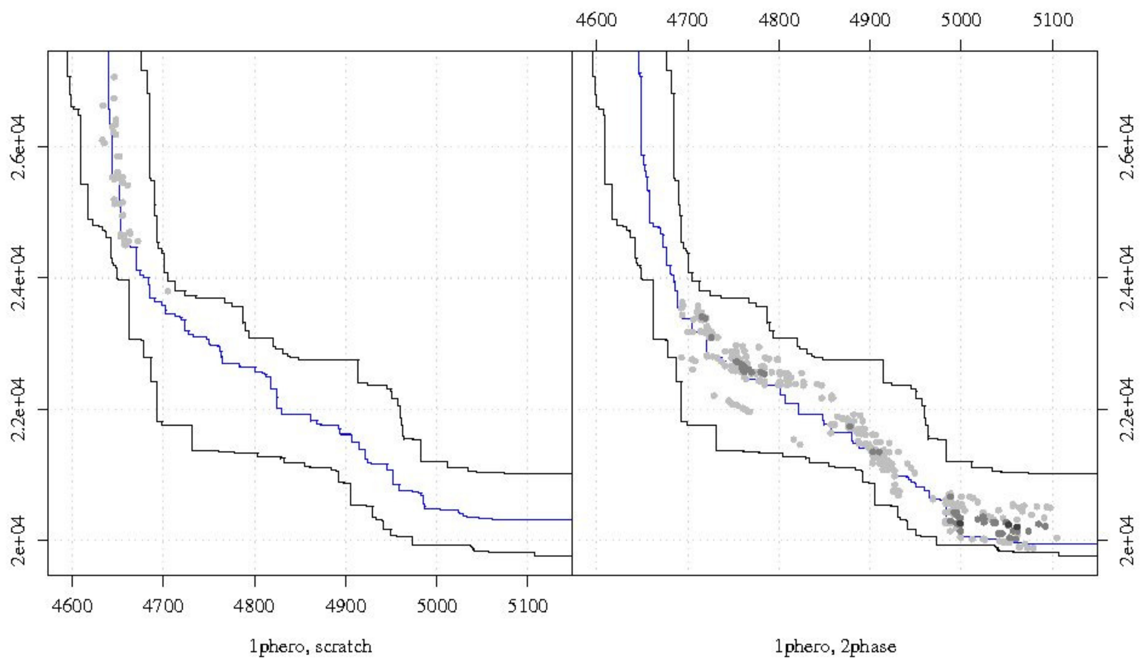


Figure 84: Differences of EAFs, *1phero scratch/1phero 2phase* (50x30-1)

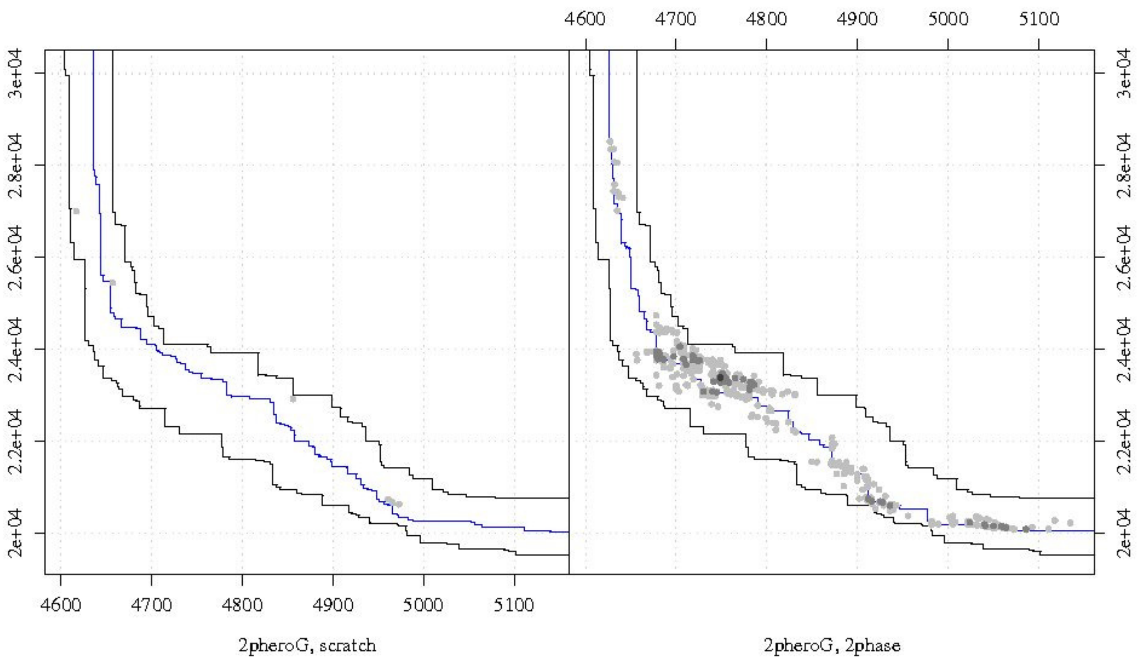


Figure 85: Differences of EAFs, *2pheroG scratch/2pheroG 2phase* (50x30-1)

λ = 1□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
(1pheroscratch/1phero 2phase)	42%	h_0 not rejected
(1phero 2phase/1phero scratch)	48%	
(2pheroG scratch/2pheroG 2phase)	40%	h_0 not rejected
(2pheroG 2phase/2pheroG scratch)	51%	
(2pheroL scratch/2pheroL 2phase)	31%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroL scratch)	62%	

Table 48: Results of a comparison *scratch/2phase* (50x30-2)

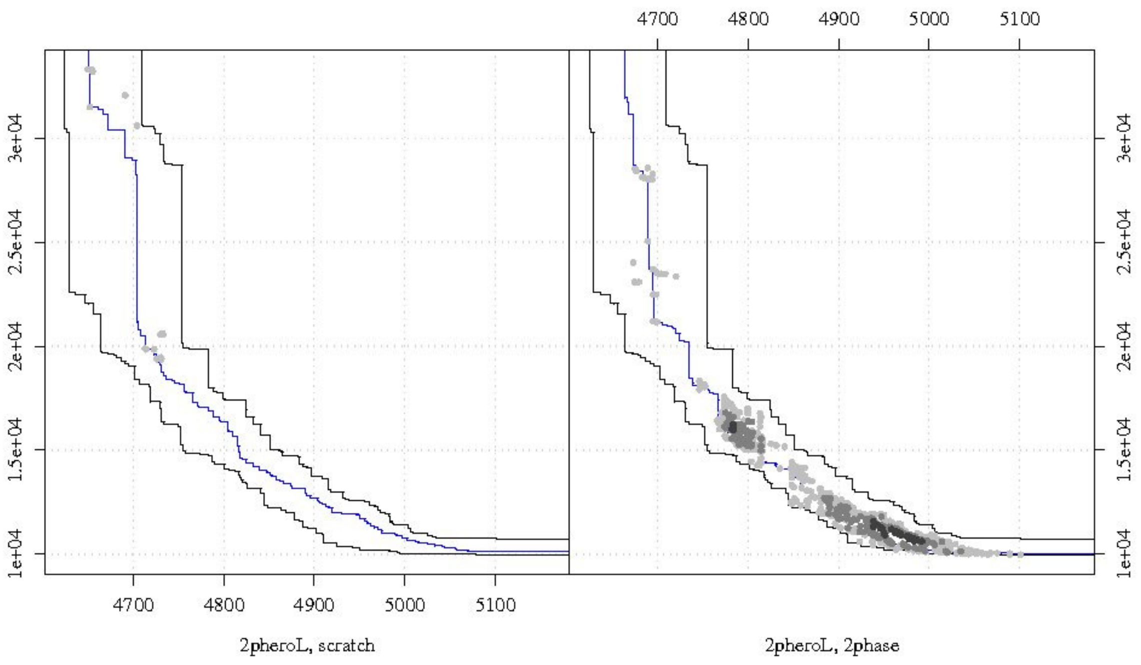


Figure 86: Differences of EAFs, *2pheroG scratch /2pheroG 2phase* (50x30-2)

Summary of the observations on the *scratch* and the *2phase* approach,

For the two instances 50 jobs, 10 machines, we can observe that:

- *1phero 2phase* approach seems to be more performing than *1phero scratch*
- There is no significant differences in using *2pheroG scratch* or *2pheroG 2phase*. This is probably due to the fact that global update makes the process converge quickly on a solution in both cases. Thus starting from the solution found in the previous iteration does not provide supplementary useful information
- *2pheroL 2phase seems to be* at least better than *2pheroL scratch* in this capability to find solutions with a small makespan, but is sometimes better in all the regions of the objective space.

For instances with 50 jobs and 30 machines, the observations differ:

- For *1phero* approach, things are not clear even if the *2phase* approach seems to be slightly better in finding solutions in the right region of the objective space

The only case where a comparison is possible for *2pheroG* provides the same observation than for 2pheroL. *2phase (G or L)* approach are at least better for solutions in the middle and in right part of the front than *2phero (G or L) scratch.*

9 Comparison *global* – *local* strategy

We present here the results of a comparison we made for the four instances to determine when one strategy is preferable to the other; the different comparisons will be presented instance by instance. Before the summary of the results, we will also presents some results showing the influence of the type of direction changes on the results of the comparison. The Table 49, 50, 51 and 52 present the results for respectively the 50x10-1, 50x10-2, 50x30-1 and 50x30-2. Among these tables, only one situation of dominance appears. In the Table 50, we observe that *2pheroG scratch* dominates *2pheroL scratch*. Nevertheless, in some situations, we can use the plot of the differences of two EAFs to compare two configurations.

λ = 1□ 0 - (50x10-1)	I_C	Kolmogorov-Smirnov test
(2pheroG scratch/2pheroL scratch)	65%	The two attainment surfaces differ somewhere
(2pheroL scratch/2pheroG scratch)	33%	
(2pheroG 2phase/2pheroL 2phase)	52%	h_0 not rejected
(2pheroL 2phase/2pheroG 2phase)	39%	

Table 49: Results of a comparison *global/local* strategy (50x10-1)

In the Figure 86, we can observe that *2pheroG scratch* seems to perform better that *2pheroL scratch* for solutions with high makespan quality. For the rest of the objective space, no preference really appears.

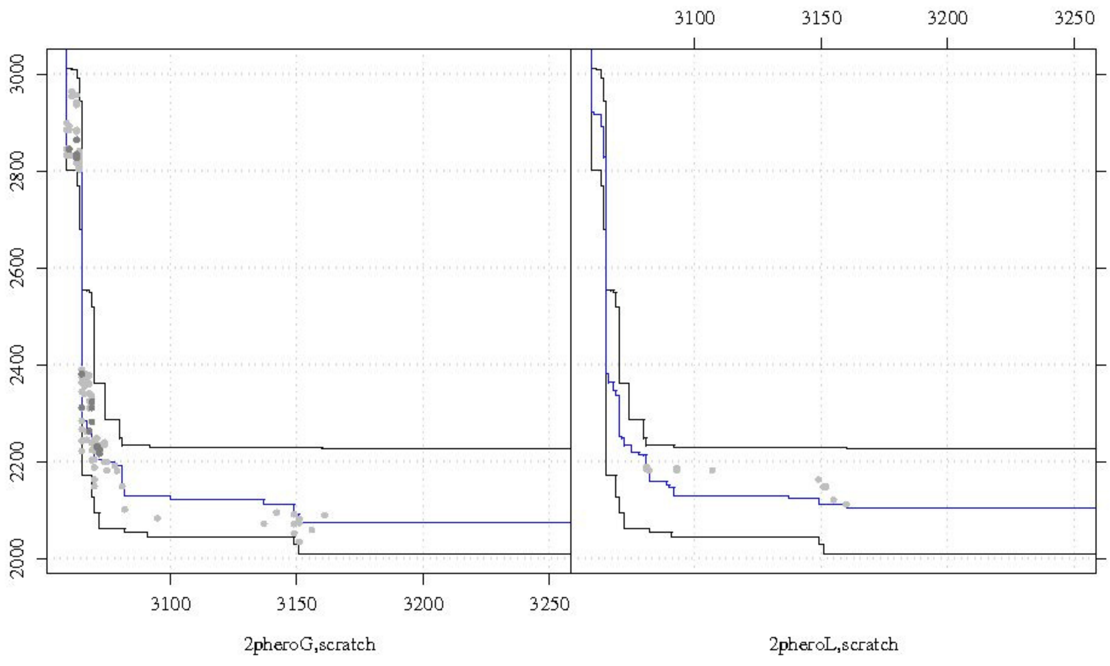


Figure 87: Differences of EAFs, *2pheroG* scratch/*2pheroL* scratch (50x10-1)

λ = 1□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(2pheroG scratch/2pheroL scratch)	77%	The two attainment surfaces differ somewhere
(2pheroL scratch/2pheroG scratch)	14%	
(2pheroG 2phase/2pheroL 2phase)	52%	h_0 not rejected
(2pheroL 2phase/2pheroG 2phase)	42%	

Table 50: Results of a comparison *global/local* strategy (50x10-2)

λ = 1□ 0 - 50x30-1	I_C	Kolmogorov-Smirnov test
(2pheroG scratch/2pheroL scratch)	39%	The two attainment surfaces differ somewhere
(2pheroL scratch/2pheroG scratch)	52%	
(2pheroG 2phase/2pheroL 2phase)	29%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroG 2phase)	60%	

Table 51: Results of a comparison *global/local* strategy (50x30-1)

For the instances with 50 jobs ad 30 machines, figures 88, 89, 90, 91 show that *2pheroL* performs most of time better than *2pheroG* in the middle of the objective space. By contrast, *2pheroG* performs sometimes better than *2pheroL* for the extreme regions, especially the left upper corner.

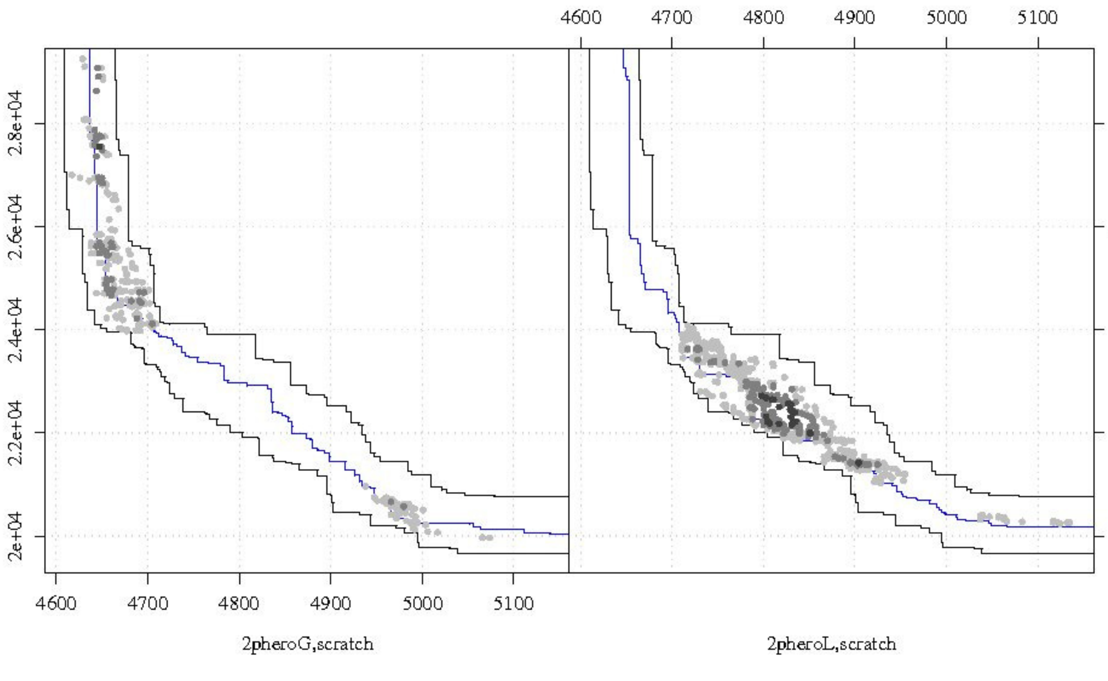


Figure 88: Differences of EAFs, *2pheroG scratch/2pheroL scratch* (50x30-1)

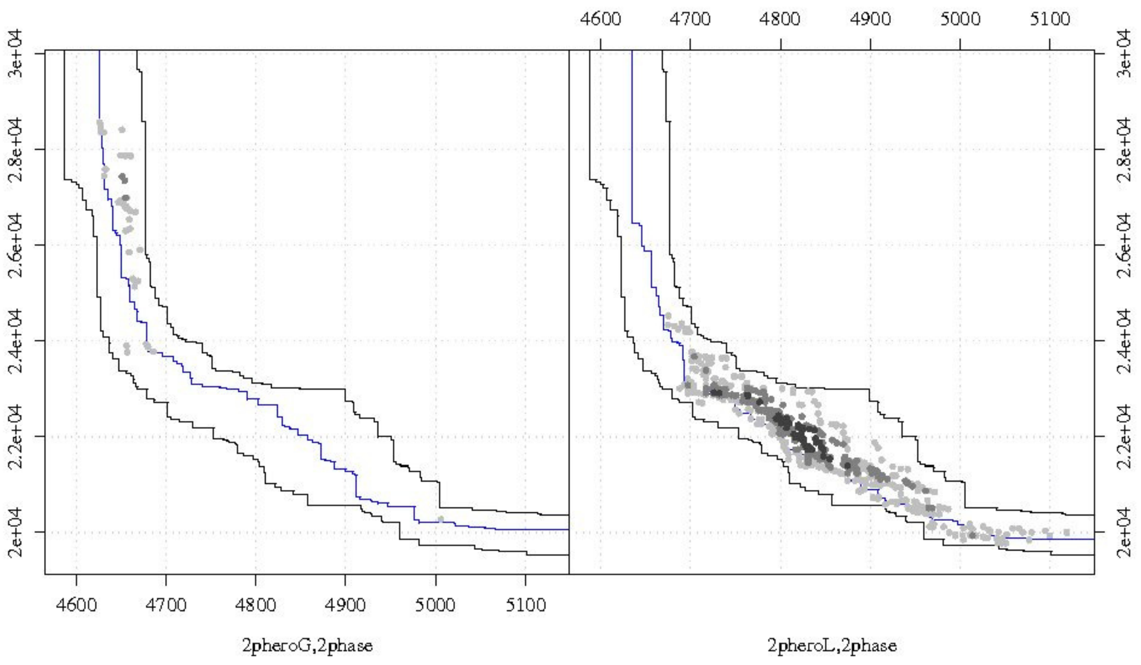


Figure 89: Differences of EAFs, *2pheroG 2phase/2pheroL 2phase* (50x30-1)

λ = 1□ 0 - 50x30-2	I_C	Kolmogorov-Smirnov test
(2pheroG scratch/2pheroL scratch)	53%	The two attainment surfaces differ somewhere
(2pheroL scratch/2pheroG scratch)	49%	
(2pheroG 2phase/2pheroL 2phase)	31%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroG 2phase)	62%	

Table52: Results of a comparison *global/local* strategy (50x30-2)

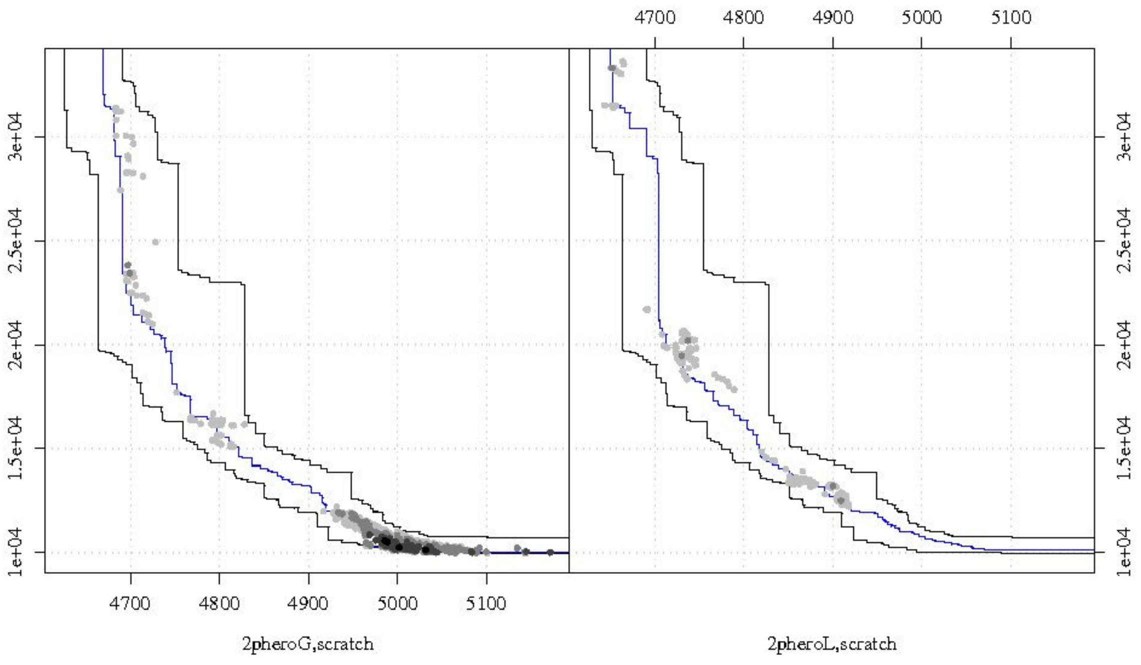


Figure 90: Differences of EAFs, *2pheroG scratch/2pheroL scratch* (50x30-2)

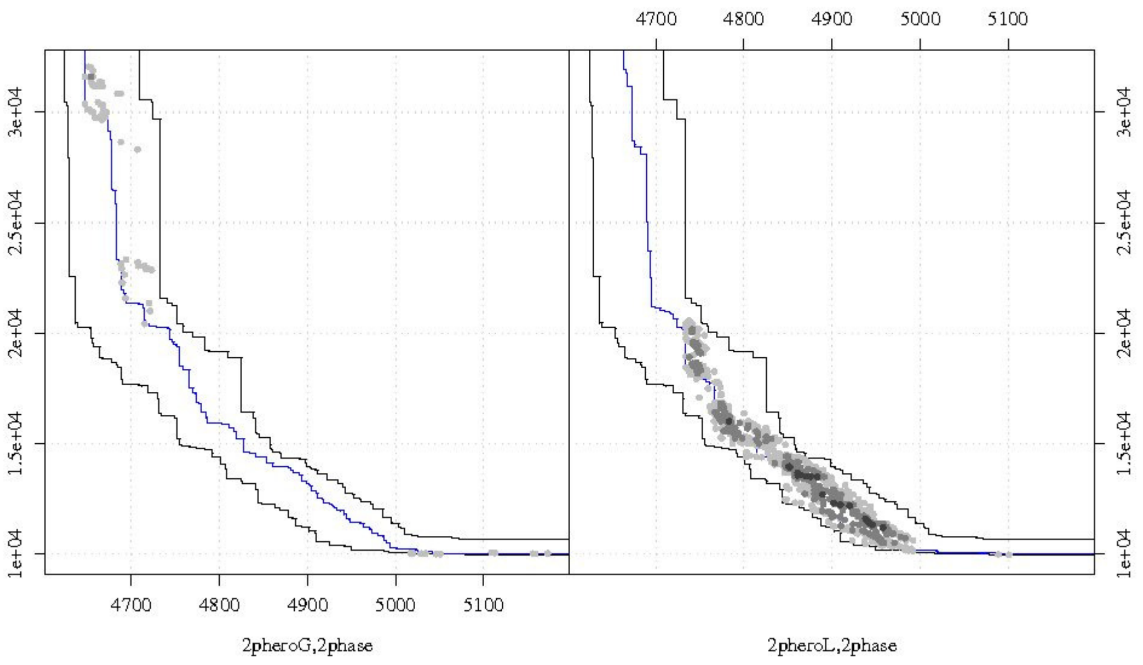


Figure 91: Differences of EAFs, *2pheroG 2phase/2pheroL 2phase* (50x30-2)

The small tables 53, 54, 55 and 56, 57, 58 present the results for comparisons *global-local* strategy for different direction changes. In these tables, two situation of clear preference appears in the Tables 56 and 58.These two results are contradictory. In the first case *2pheroG 2phase* is better than *2pheroL 2phase* for a type of direction changes $\lambda = 0...1$ and in the second case *2pheroL 2phase* is better than *2pheroG 2phase* for a type of direction changes $\lambda = 1...0...1$. The Figure 91 also indicates that the way the directions change has its importance in the performance of the algorithm. The result of this comparison on the (50x10-2) with $\lambda = 1...0...1$ is different than the result obtained with $\lambda = 1$□ 0.

λ = 0 ↗ 1 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(2pheroG 2phase/2pheroL 2phase)	40%	h_0 not rejected
(2pheroL 2phase/2pheroG 2phase)	53%	

Table 53: Results of a comparison *global/local* strategy for λ=0-1 (50x10-2)

λ = 0□ 1□ 0 - (50x10-2)	I_C	Kolmogorov-Smirnov test
(2pheroG 2phase/2pheroL 2phase)	38%	h_0 not rejected
(2pheroL 2phase/2pheroG 2phase)	54%	

Table 54: Results of a comparison *global/local* strategy for λ=0-1-0 (50x10-2)

λ = 1□ 0□ 1 - 50x10-2	I_C	Kolmogorov-Smirnov test
(2pheroG 2phase/2pheroL 2phase)	25%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroG 2phase)	69%	

Table 55: Results of a comparison *global/local* strategy for λ=1-0-1(50x10-2)

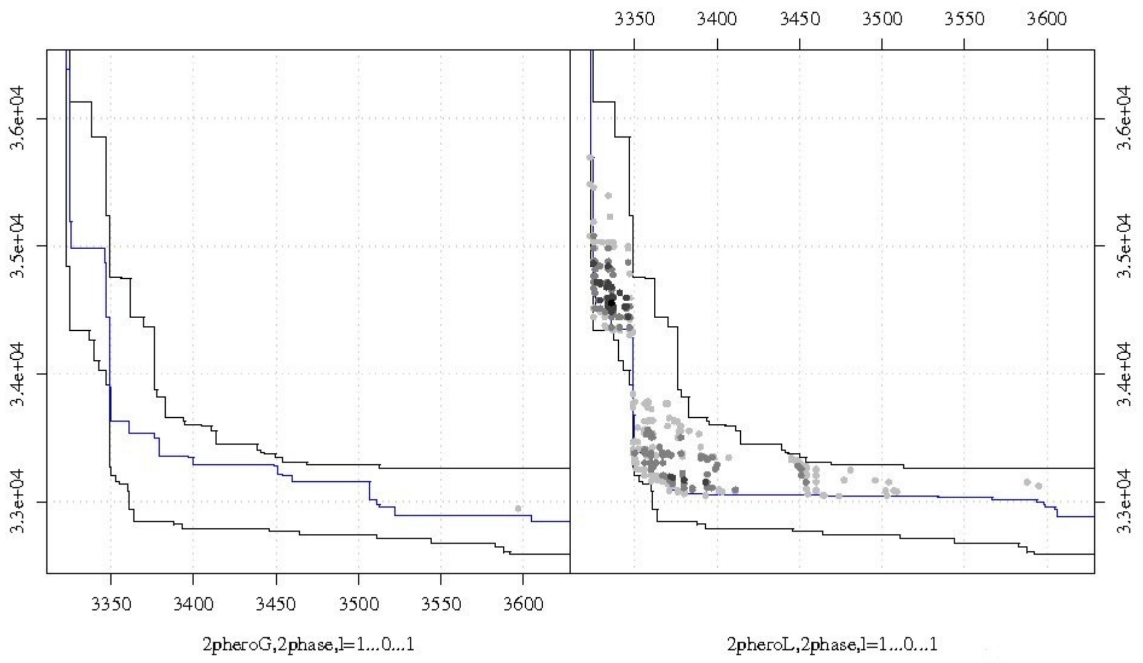


Figure 92: Differences of EAFs, *2pheroG 2phase/2pheroL 2phase* for λ=1-0-1 (50x10-2)

$\lambda = 0...1$ - 50x30-2	I_C	Kolmogorov-Smirnov test
(2pheroG 2phase/2pheroL 2phase)	70%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroG 2phase)	27%	

Table 56: Results of a comparison global/*local* strategy for λ=0-1 (50x30-2)

λ = 0□ 1□ 0 - 4th instance	I_C	Kolmogorov-Smirnov test
(2pheroG 2phase/2pheroL 2phase)	52%	h_0 not rejected
(2pheroL 2phase/2pheroG 2phase)	43%	

Table 57: Results of a comparison *global/local* strategy for λ=0-1-0 (50x30-2)

λ = 1□ 0□ 1 - 50x30-2	I_C	Kolmogorov-Smirnov test
(2pheroG 2phase/2pheroL 2phase)	15%	The two attainment surfaces differ somewhere
(2pheroL 2phase/2pheroG 2phase)	78%	

Table 58: Results of a comparison *global/local* strategy for λ=1-0-1 (50x30-2)

Summary of the observations on the *global* and *local* strategy.

For the two instances 50 jobs, 10 machines, we can observe that:

- *2pheroG scratch* seems to be more performing than *2pheroL scratch* at least if the decision maker prefers solutions with a small makespan
- There is no indications that applying *2pheroG 2phase* or *2pheroL 2phase* to the problem gives better solutions for one region or another in the objective space
- Results may change following the direction changes used for the aggregation.

For the two instances 50 jobs, 30 machines, the observations are different, we can observe that:

- *2pheroL* seems to be more capable of finding good solutions in the middle region of the objective space
- *2pheroG* seems to be more performing in the extreme regions of the objective space, especially for the left upper corner
- Results may change following the direction changes used for the aggregation.