Université de Tunis
Institut Supérieur de Gestion
LARODEC Laboratory

Resource-Constrained Project Scheduling Problem: A Multi-Objective model with contingency

Elaborated by: Olfa Dridi Ben Zekri

Co-advised by: Fouad Ben Abdelaziz
Saoussen Krichen

june 2007

To the memory of my Grand Mother,

To my dear parents,

To my husband Riadh,

To my sisters,

To all my family and my friends.

Acknowledgements

I am deeply indebted to my supervisors Professor Fouad Ben Abdelaziz and Mrs Saoussen Krichen for their support, encouragement and valuable discussions concerning this research. I greatly benefited from their guidance during this work.
I wish to thank all person who provided me information, advice, criticism and encouragement.

Contents

List of Tables

List of Figures

General introduction

Many real-life decision situations involve multiple objectives that should be simultaneously optimized. These objectives are generally conflicting and heterogeneous [22]. Such a decision problem is called a Multi-Objective Optimization problem (MOP). This kind of problems has increased tremendously in almost various domains such as industry, engineering and scheduling. To solve MOPs, classical methods reformulate them into a single objective optimization problem by using an aggregation function [15]. This aggregation requires an interactive approach, in other words, allows the decision makers (DMs) to incorporate his preferences and priority for each objective into a decision process [28]. The optimization of such problem provides a single optimal solution. But, it is often not advisable to reduce the MOP to a scalar optimization. It is preferable to tackle them as MOP. In these cases not a single solution can be found but a set of efficient solutions or pareto-optimal solutions.

The majority of the multi-objective combinatorial optimization problems have conflicting criteria and hard constraints that should be optimized and satisfied at the same time. Therefore, they are classified as *NP*-hard problems : they need an exponential time to be solved. So, it is impossible or very difficult to solve them in a reasonable time. In this sense, the feasible way to solve this *NP*-hard problem is the use of heuristic approaches, i.e. approximate algorithm, to obtain potentially non-dominated solutions. There are two type of approximate algorithms : *construction algorithms* as

greedy algorithm and *local search algorithm*, and *improvement algorithms* as *the Tabu Search algorithm*.

In the field of combinatorial optimization problems, we turn our attention to the well-known Resource-Constrained Project Scheduling Problem (RCPSP) which is one of the most important problems in project scheduling domain. It is a generalization of the job-shop problem one of the most studied problem in combinatorial optimization theory [3] [4]. There are many variants of the RCPSP; multi-mode, single mode, with non pre-emptive or pre-emptive resources, renewable or non-renewable resources [38].

The classical RCPSP tries to schedule activities in order to minimize the makespan of the project without violating precedence and resource availability constraints. An overview of heuristic and exact methods can be found in [24][23]. However, in real life situations scheduling problems require the consideration of several objectives that are to be optimized simultaneously. Such objectives are conflicting by nature as the time duration of the project and the cost of the resources used to perform the activities of the project.

The literature on multi-objective RCPS problem is scant. However, in single objective case, this problem was extensively studied. That's why, in the existing literature there are a large variety of heuristic and exact methods proposed to solve each variants of uni-objective RCPSP. Some researchers consider the problem with single mode aspect, others consider the problem with multiple mode where activities can be pre-emptive or not. As : Merkle et al. [33] [32], who present the first application of Ant System metaheuristic to solve the single mode and uni-objective RCPS problem, Demeulemeester et al. [10] developed an exact method for solving single mode and uni-objective RCPSP by applying an implicit enumeration procedure of the Branch-and-Bound. Genetic Algorithm has been also applied to uni-objective RCPSP [37]. About the multi-objective RCPS problem, Belfares et al. [4] presented a Tabu Search method that consists in modelling the course of actions planning as a multiple mode RCPSP. Belfares et al. [4]

proposed a progressive resource allocation method based on the Tabu Search metaheuristic and multi-objective concepts. Al-Fawzan and Haouari [2] developed a method based on Tabu Search to solve the single mode RCPSP with two objectives: makespan and robustness.

It is note worthy that real world environment is usually characterized by uncertain and incontrollable informations. Therefore, it seems that classical approaches are inefficient. In fact, they neglected the state of the environment and its instantly evolution. Hence, the production of contingent plans is necessary in the planning process. In this research, we introduce and define the concept of contingency planning and we develop a multi-objective RCPS model with a multi-mode feature and contingency. For that, we propose a new method based on Ant System metaheuristic and multi-objective concepts to raise the issue of the environment uncertainty and to schedule activities.

This thesis is framed as follows:

Chapter 1 deals with the definition and the classification of the RCPS problems.

Chapter 2 draws upon the definition and the utility of contingent plans in the planning process and provides a description of some categories of planning under uncertainty.

Chapter 3 contains an overview of the main metaheuristics used to tackle the multi-objective combinatorial optimization problems.

In chapter 4 we present our contribution and consists in developing a new approach to handle the aspect of uncertainty in the planning process. We finish by conclude this thesis.

Chapter 1

Multi-Objective Resource-Constrained Project Scheduling problem: A survey

1.1 Introduction

Facing the globalisation and the growing number of international competitors, and in an attempt to promote development and to avoid failure, the manager should predict a good plan. Hence, in project scheduling problems planning and scheduling activities are viewed profoundly important to generate plans and to maximize the utilization of scarce resources. Thus, finding a feasible and efficient plan is a considerable challenge.

In this respect, the well known Resource-Constrained Project Scheduling (RCPS) represents the most important problem in project scheduling. There are many variants of this problem; multi-mode, single mode, with non pre-emptive or pre-emptive resources, renewable or non-renewable resources [38]. The classical RCPSP tries to find suitable solution in order to op-

timize the makespan, i.e. to find the best resource allocation for a given set of activities/tasks. But, in real-life the scheduling problem requires the consideration of several objectives, generally conflicting, as the time duration of the project and the cost of the resources used to perform the activities of the project. Thus, the project manager faces the problem of finding good feasible schedules taking into account multiple objectives and not a single objective. In this context, the notion of optimality is replaced with the notion of efficiency and instead of a single optimal solution we deal with a set of trade-offs called efficient solutions or Pareto-optimal solutions. The literature on multi-objective RCPS problem is scant, hence, there are few papers about it.

The multi-objective RCPS problem is considered as an *NP*-Hard problem [1]. For that, metaheuristics are applied. In the literature we can find the Tabu Search algorithm [5] [4], The Ant Colony Optimization algorithm [16].

This chapter is structured as follows: in the first part we will define the problem. Next, we will present the mathematical formulation and provide an illustrative example. Then, we will define the classification of the RCPS problem. Finally, we will present an instance of multi-objective RCPSP.

1.2 Description of the problem

1.2.1 Definition of the RCPS problem

Resource-Constrained Project Scheduling problem can be defined as a set of resources with limited capacities to be assigned to a set of tasks or activities to be performed. The tasks are interrelated by precedence constraints that enforce them to be started only if their predecessors were accomplished. The structure of this problem can be depicted as an acyclic graph where nodes represent tasks and arcs represent precedence relations. Thus, the objective of the RCPS problem is to find a suitable assignment of the resources to the

tasks such that the constraints are fulfilled and in order to optimize some predefined goals or objectives [4] [33]. There are several objectives to be optimized. We can enumerate some of these objectives, stated in the literature as below [21] [4] [26] :

- The minimum makespan: is the most used and applied objective of a general project scheduling problem. It reduces the time to complete the entire project, in other words, is defined as the total time span between the start and the end of the project;
- The cost minimization : has attracted a lot of attention due to its practical significance. It consists in reducing the cost of the resources used and include the case where tasks may be performed in several mode resulting in different costs;
- The quality maximization: maximize the quality of the project, i.e improve the quality of the product and services.

1.2.2 Definition of a Course Of Actions (COAs)

We define a project M as a set of n ordered tasks T related by a set of predecessors P_i, with $i = 1, ..., n$. To be executed, each task needs a combination of resources c. The couple (Task t_i, Resource c) constitute an action denoted by a_{ic}. Hence, a COA can be represented by a set of tasks to be accomplished, in such a way the constraints of precedence relationships between tasks and resources availability are fulfilled [4] [21]. A COA = (a_{ic}, P_i), the COA can be modelled as follows in figure (1.1):

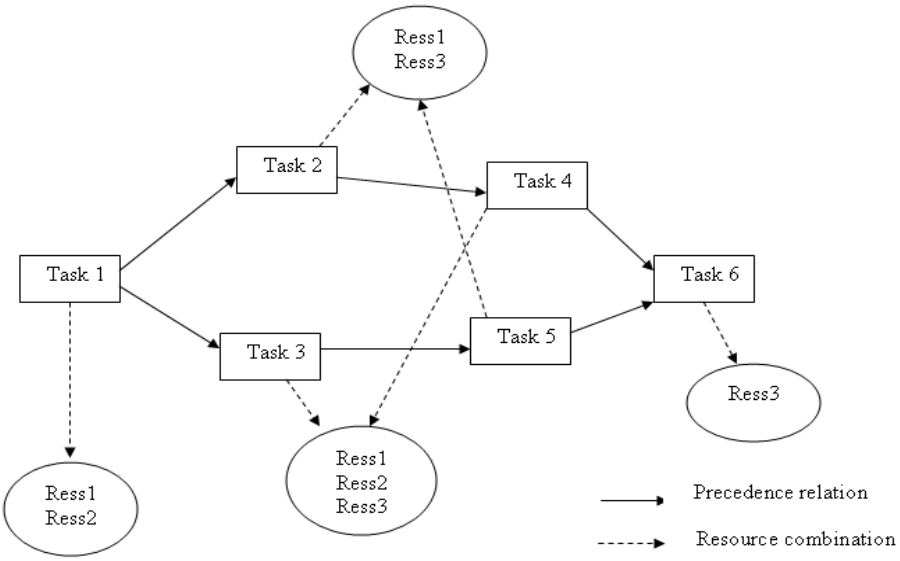

Figure 1.1: The course of actions

1.2.3 Definitions of planning and scheduling

In the literature, the terms planning and scheduling are binded. Thus, the distinction between them are usually ambiguous. In other words, scheduling is the determination of suitable allocation of resources to tasks over time and taking into account the constraints of the problem. As for planning, is the generation of a sequence of tasks in order to accomplish some goals and respecting the constraints of the problem [31].

1.2.4 Notation

T	:	the set of tasks.
K	:	number of resource.
n	:	number of task.
t_i	:	the task i.
P_i	:	the set of predecessors of task i.
R_j	:	the available quantity of resource of type j.
r_j	:	the resource of type j.
$r_{i,j}$	:	the quantity of resource j, $j \in K$, required to task i.
d_i	:	the duration of task i.
s_i	:	starting time of task i.
f_i	:	finishing time of task i with $f_i = s_i + d_i$.
CF_j	:	the fixed costs of resource of type j.
CV_j	:	the in-use costs of resource of type j.
c_j	:	the cost of the resource j.
x_{ij}	=	$\begin{cases} 1 & \text{if } t_i \text{ is realized by resource } j \\ 0 & \text{otherwise} \end{cases}$

1.2.5 Mathematical formulation

Generally, the most decision problems are multiobjective in nature and we have to optimize several objectives at the same time. The most frequently objectives used in the scheduling and resource allocation problems are the makespan (C_{max}), the net present value, the cost, reliability, probability of success.

The multiobjective RCPSP is defined as the way to assign a set of tasks to a set of resources with limited capacity in order to optimize M objectives simultaneously.

Thus, the mathematical formulation can be stated as follows:

$$\begin{aligned} &Min \;\; C_{max} \\ &\quad \vdots \\ &Min \;\; Cost \end{aligned} \tag{1.1}$$

Subject to:

$$\sum_{i=1}^{n} x_{ij} \; r_{ij} \leq R_j, \quad j = 1, \ldots, K \tag{1.2}$$

$$s_i \;\geq\; Max_{p \in P_i} \; (s_p + d_p) \tag{1.3}$$

$$x_{ij} \in \{0, 1\} \tag{1.4}$$

Equations (1.1) are the objective functions to optimize. The resources availability constraint (1.2), means that each task i is performed by r_{ij} quantity of resource j that can't exceed the available quantity R_j. Equation (1.3) is the predecessor constraint, guarantees that each task i starts if all their predecessors P_i have been finished. Constraint (1.4) defines the decision variables, where:

$$x_{ij} = \begin{cases} 1 & \text{if resource } j \text{ contributes to the realization of task } i. \\ 0 & \text{otherwise.} \end{cases}$$

1.2.6 An illustrative example

We have a mission with n tasks $n = 4$ related by a set of predecessors P_i and two renewable resource type r_1 and r_2 which are available with $r_1 = 4$ units and $r_2 = 5$ units each period. To be executed, each task needs a resource allocation. The solution consists in finding a feasible allocation of available resources to the set of tasks such that the makespan is minimized, and the constraints of precedence relationships between tasks and resources availability are fulfilled. Hence, we have to find the vector ν of the finishing time of all the tasks $\nu = (f_1, \ldots, f_4)$.

In this respect, each task $i \in T$ has a duration time d_i and resource requirement $r_{i,j}$. We suppose that P_i be the set of predecessors of task i, then, task i must not be started before all its predecessors are finished, i.e $f_p \leq s_i$ if $t_p \in P_i$. We assume that task 1 must be performed by the r_1, task 2 and 3 can be performed by r_1 or r_2 and task 4 requires r_1 and r_2. The following table summarizes the example data:

task j	P_i	d_i	r_1	r_2
1	-	3	2	-
2	1	4	3	4
3	1	2	4	3
4	2,3	1	1	4

The example can be formulated as follows:

$$Min \quad C_{max} \; = \; Max \sum_{i=1}^{4} f_i \tag{1.5}$$

subject to:

$$\sum_{i=1}^{4} r_{i,j} \leq R_j, \quad j = \{1, 2\} \tag{1.6}$$

$$f_p \leq \; f_i \; - \; d_i, \;\; i \; = \; 1, \ldots, 4; \;\; t_p \in P_i \tag{1.7}$$

$$f_i \geq \; 0 \;\; i \; = \; 1, \ldots, 4 \tag{1.8}$$

Equations (1.5) is the objective function: minimize the finish time of the project. The resources availability constraint (1.6). Constraint (1.7) enforces the precedence relationships between tasks. Constraint (1.8) defines the decision variables.

The structure of the problem can be represented as an oriented graph with tasks as nodes and precedence constraints as edges:

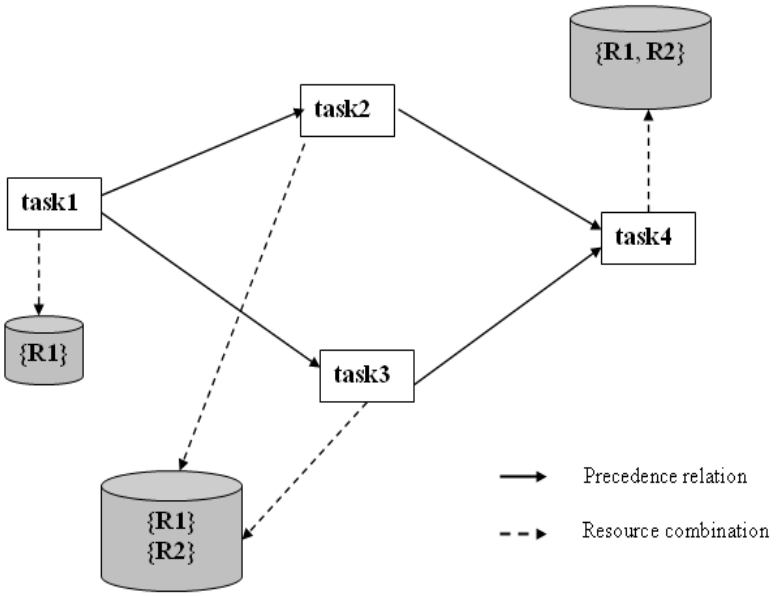

Figure 1.2: An illustrative example of the RCPS problem with $n = 4$ and $k = 2$

The objective is to construct a COAs such that the makespan is minimized:
Min $C_{max} = \sum_{i=1}^{4} f_i$.
The optimal solution $C_{max} = 8$, with the vector $\nu = (3, 7, 5, 8)$.

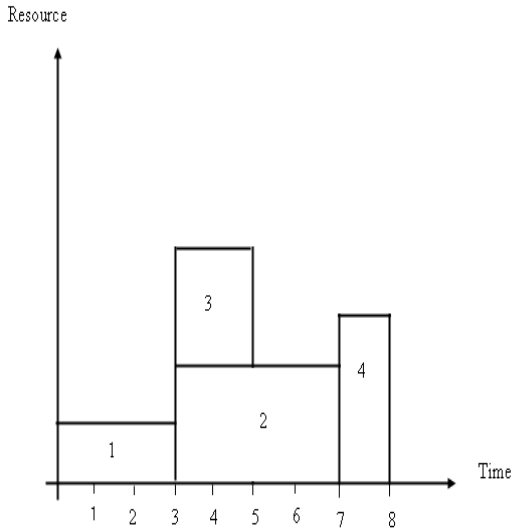

Figure 1.3: The optimal solution for the illustrative example

1.3 The classification of resource-constrained project scheduling problem

We present in this section a brief survey on the variants of the RCPS problem. We can find in the literature: the single mode, multi-mode, with non-regular objective functions, Stochastic feature, bin-packing-related RCPSP and multi-RCPSP

1.3.1 Single mode RCPSP

For the single-mode RCPSP (SM-RCPSP), each task has a single execution mode. In other words, for each activity: the processing time and its requirements for a set of resources are assumed to be fixed in advance, and only one execution mode is available [38]. The set of tasks is constrained by the

precedence relationships, for example the task i must be completed before starting the task j. we distinguish two different precedence relationships: in the first one task j can start at any time following completion of task i, in the second one task j must start within some time window following the completion of task i. We call these latter restrictions General Precedence Relationships (GPR).

1.3.2 Multi-mode RCPSP

The multi-mode RCPSP (MM-RCPSP) is defined as a RCPS where each task has to be processed in one of several modes. Each mode implies a different options in terms of cost, processing time, amount of a particular resource for achieving the task [38]. For example: a single technician can finish a given job in 2 days is mode 1, whereas, two technicians can finish the same job in 1 day is the mode 2. So, the job can be completed by either mode 1 or mode 2 with different values in terms of cost and time. There are two different classification of resources used to complete tasks: renewable or non-renewable. Non-renewable resources are exhausted after a certain amount of consumption, while renewable resources have the same amount of availability in every period.

1.3.3 RCPSP with non-regular objective functions

The regular objective function is one in which the objective function is never made worse by reducing the completion time of a job without increasing the completion time of any other job [38]. However, a non-regular objective function don't respect this property. Because the value of the objective function can actually increase by reducing the completion time of an activity, and so these are non-regular objective functions.

1.3.4 Stochastic RCPSP

In this problem, the processing time of each task/activity and the resource (cost) requirement are considered as a random variables following some probability distribution [8]. In this case, instead of minimizing the makespan, we consider the minimization of the expected makespan. Many interdependent activities are often represented in a project graph, and activity completion times are highly interdependent, the probability distribution of the total makespan is often difficult or impossible to characterize and often leading to activity independence assumptions for tractable analysis. This independence assumptions can provide misleading results in practice [8] [38].

1.3.5 Bin-packing-related RCPSP

A bin-packing problem can be described as follows: given a standard bin size and a set of items, the objective is to assign each item into a bin while minimizing the total number of bins used. A bin-packing-related RCPSP refers to a special kind of resource-constrained project scheduling problem that can be seen as analogous to a bin-packing problem. Many resource-constrained project scheduling problems can be viewed as a generalization of the basic bin-packing problem. The analogy between the bin packing and RCPS problems can be explained as follows: The resource availability represents the bin size, while a tasks resource consumption needs represent an item size. In the RCPSP, we can modelize each period as a bin into which we can pack different tasks. We can also make the following assumptions: each task takes less than one period of resource consumption and every task must be fully completed within a single day, so, minimizing makespan of this RCPSP is equivalent to minimizing the total number of bins used in an equivalent bin-packing problem [38].

1.3.6 Multi-RCPSP

In the multi-resource-constrained project scheduling problem (M-RCPSP), a job may require a set of operations, or a set of resources. For a given operation, various resources can be used in parallel. In other words, to be executed each task/job might select any one of these resources. Where successive resources are needed in series and if a task select a given resource then it is not allowed to use another resource before completing processing on the first one. These problems are often called machine-scheduling problems. In the manufacturing area, a resource is equivalent to a machine that can only process one job at any time. Machine and job scheduling characteristics can be classified to several categories [38]:

1. Identical machines in parallel.
2. Machines in parallel with different speeds: the speed of machine i is v_i for all jobs.
3. Unrelated machines in parallel: speed of machine i for job j is v_{ij}.
4. Flow shop: machines are in series; every job has to be processed on each of the machines in the same sequence.
5. Job shop: a job can visit any subset of the machines in any order.

1.4 Multi-modes and multi-objective RCPSP

The multi-objective RCPSPs are considered *NP*-hard optimization problems and can be solved, in a polynomial time, by using a heuristic procedure. In the literature, there are various categories of RCPSPs. Therefore, in recent years the number of algorithms proposed to solve the uni-objective RCPS are increased tremendously. About the multi-objective RCPS problems there are few papers in the literature. For instance, Guitoni et al. [5] [4] presented a method consists in modelling the military Operational Planning Process

(OPP) as a multiple modes RCPSP. Belfares et al. [4] proposed a Progressive Resource Allocation (PRA) method based on the Tabu Search metaheuristic to solve the multi-mode RCPS problem with three objectives: minimize the cost, maximize the reliability and maximize the probability of success. The OPP is composed of five stages:

1. Initiation stage: reception of a mission (mission is a set of tasks).
2. Orientation stage: the commander proceed to analyse the mission and prepare the aimed to realize.
3. Course of action development stage: the staff generate the planning or the Course Of Actions (COAs) suited to the commander's guidance and intent, i.e. the process of planning consists in first of all to identify the tasks to realize, their requirement in resource and their precedence relationships.
4. Plan development stage: analysis and comparison of the COAs.
5. Plan review stage: the choose of the best COA.

In fact, due to it fundamental role in the generation of a good COAs according to the commander's guidance and estimates, the operational planning process (OPP) should be efficient. Therefore, Belfares et al. [4] focused on the COA development stage. During this stage, the staff should identify the assigned tasks to perform the mission. To be executed, each task needs resources and assets allocation. The couple (task, resource) constitute an action.

A COA can be represented by a set of tasks to be accomplished, in such a way the constraints of precedence relationships between tasks and resources availability are fulfilled. The COA planning can be modeled as a multiple mode RCPSP (Is defined in the previous section). In their work, Belfares et al. [4] are concerned with the multi-objectives aspect of a multiple mode RCPSPs. The aim is to find suitable resource allocation for given COA

schedule [5]. So, the problematic of their paper is: Given a set of limited resources/capabilities, what is the best allocation for a given task schedule according to several objectives?

1.4.1 Resolution of the multi-mode RCPSP with the PRA procedure

To find a good allocation of resources, Belfares et al. [4] propose a Progressive Resource Allocation methodology based on the tabu search method and the pareto optimization technique. This latter strategy has attracted the interest of researchers and is deeply used in the multicriteria framework [4]. Furthermore, the optimization method PRA is based on a posteriori approach i.e. various solutions are found and then, the decision maker (the commander) select the most suitable. The goal is to find the largest number of well diversified efficient solutions. For that, an iterative filtering or/and a choice process might then be carried out with the decision maker to select the most suitable solutions (COA planning). Belfares et al. [4] used the Multi-criteria Filtering Procedure (MFP) and the Dominance Rule (DR) to filter and choose the solutions generated by their proposed procedure. We present an overview of the MFP: let A a subset of non-dominated solutions and let Card(A) the user-defined cardinality of this subset. The MFP:

- Step 1: The Disjunctive Method: retains all the solutions that score a maximal value on at least one objective F_i.
- Step 2: Number of solutions retained $\prec$ Card(A). If not go to Step 4.
- Step 3: The Conjunctive Method: applied to reach Card(A).
- Step 4: End of MFP.

We can modeled the PRA procedure as shown in figure (1.4):

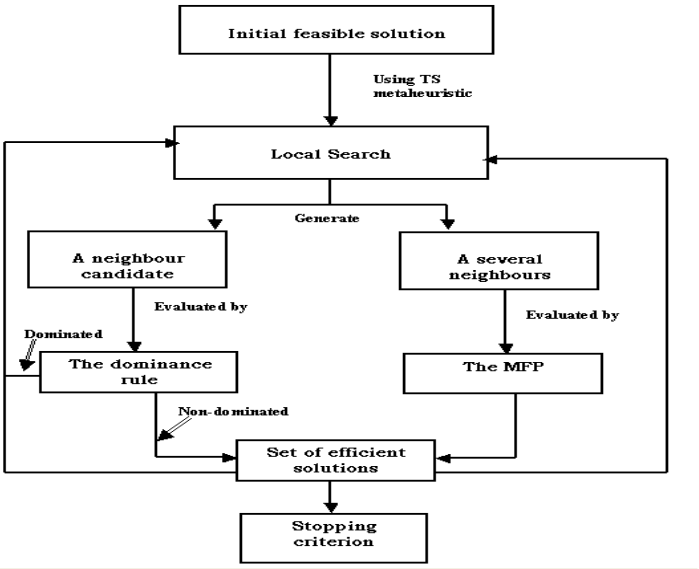

Figure 1.4: The Progressive Resource Allocation procedure

1.4.2 Main features of the PRA approach

PRA (Progressive Resource Allocation) approach is proposed by Belfares et al. [4] to find an efficient Course Of Actions planning. The PRA method uses the concepts of Tabu Search (TS) metaheuristic. Due to its simplicity and efficiency to solve *NP*-hard optimization problems, TS was broadly used by researchers in different realms. The TS was well suited for tackling the RCPSPs. The adaptation of this approach to the PRA method is presented as follows:

The PRA method deals with management of tabu list. Belfares et al. [4] define two tabu list: the first list is the tabu resources, is a static list in which we prevent the use of a generic resource twice in the same schedule and the second list is the tabu tasks, is a dynamic list due to aspiration criteria, in which the assignment of resource $r_{i,k}$ to task t_j is not allowed if this combination presents a dominated COA solution.

Definition of the neighborhood: is all the possibilities of allocating specific resource of a generic set. The definition of neighborhood based on the best-scored allocation. Links between the PRA method and the TS approach can be stated in this table (1.1):

TS	**PRA method**
TS is a local improvement method.	the improvement is done on each objective to escape local optimum.
Accept sometimes a non-improving solutions in order to escape local optimum.	Non-dominated solutions are retained.

Table 1.1: Links between TS and PRA method

Local search in the neighborhood is a succession of intensification and diversification phases.
The intensification phase: explore profoundly the neighbors (the set of generic resource) of the each current solution. And this is defined as a direction of search.
The diversification phase: with each direction, we improve the multi-objective environment of search with the examination of the objective functions. It is outlined in figure (1.5):

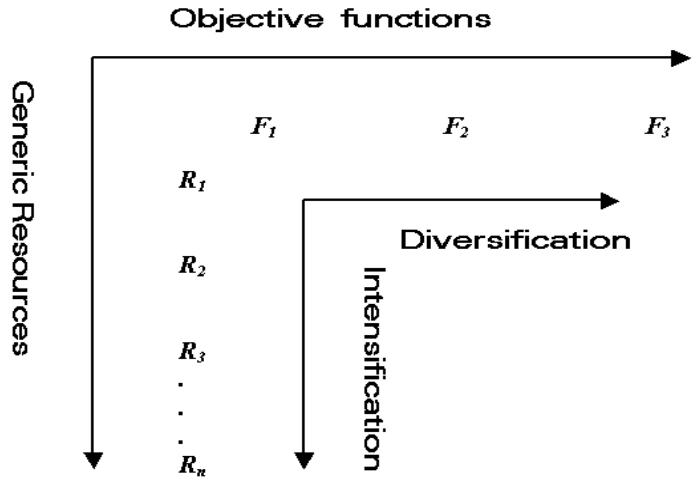

Figure 1.5: The intensification and diversification stages

1.4.3 Description of the PRA algorithm

The notation:

E_{GR}	:	a set of generic resources.
E_F	:	a set of objective functions.
R_k	:	generic resource of type k.
F_i	:	an objective function.
r_{ik}	:	a specific resource i of a generic resource of type k.

Description of the algorithm:

The Progressive Resource Allocation algorithm contains five steps:

Step 1: choose randomly the generic resources from E_{GR}.

Step 2: we remove the specific resource from the current solution and place it in the depot to be available for a new allocation.

Step 3: allocation of the best scored resource according to the objective function chosen from E_F.

Step 4: check if all objectives are considered for the current solution in order to diversify the solutions.

Step 5: intensify the search by checking if all the generic resources are considered.

The PRA is illustrated as follow :

- Initialization:
  - Consider a feasible solution
- Iterative process:
  At iteration i
  Step 1:
  - Choose randomly a generic resource R_k from E_{GR}, E_{GR}=$E_{GR}\backslash\{R_k\}$
  Step 2:
  - Remove from the current solution all $r_{ik} \in R_k$ and place them in their corresponding depot
  Step 3:
  - Choose objective F_i from E_F, E_F=$E_F\backslash\{F_i\}$
  - Construct a new solution accordingly
  Step 4:
  - Check if all the objectives are performed $E_F = \emptyset$. Otherwise return to step 3 (diversification)
  Step 5:
  - Check if all the generic resources are changed: $E_{GR} = \emptyset$. Otherwise

return to step 1 (intensification)

1.5 Conclusion

Multiobjective RCPS problem is one of the most difficult optimization problem. This difficulty arises from the fact that we have to optimize several objectives simultaneously, with some of them are generally conflicting and heterogeneous. In fact, the RCPSP modelize situations where we have to generate plan and to optimize the use of scarce resources.
We presented in this chapter the definition and the classification of the RCPSP.
Thus, the manager in the real world should take into consideration the aspect of environment. That's why, planners must predict uncertain outcomes to success their plans and to avoid potential damage. In this context, the contingency planning is viewed primordial to catch the instantly evolution of the environment. In the next chapter we propose a survey of contingency planning.

Chapter 2

Contingency planning

2.1 Introduction

Classical plans assume complete and accurate information of the planning process and about the state of the world. Actually, the environment is characterized by incomplete and uncertain information due to presence of unpredictable and incontrollable events. In this respect, the classical approach will be inefficient and researchers should take into consideration the aspect of the environment within the planning process. To handle and to surround situations where there are incomplete and uncertain information about the state of the environment, we use contingency planning. The contingency planning consists in providing flexible plan able to face unforeseen events [7] [36].

In this chapter, we start by stating the definition of the contingency planning and its necessity for the project scheduling. Next, the classification of contingency plans will be presented. Finally, we will propose an overview of contingency plans.

2.2 Definition of contingent plan

The planning process consists in generating feasible course of action the so-called 'plan' which its execution would allow the accomplishment of the tasks. The plan to be executed can lead to several results and this stems mainly from not only the unpredictable events but also the imperfection of the agent executer beside the influence of competitors. Hence, the planning managers should take into consideration environmental information, natural and man-made state, resources constraints,...to develop a comprehensive and flexible plan. So, contingent plan provides flexibility to face unforeseen events due to the evolutive environment [7] [21]. The contingent plan can be modeled as follows in figure (2.1):

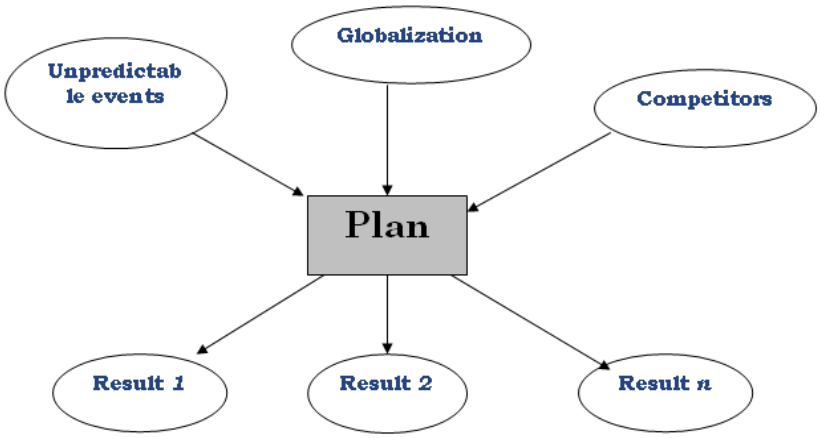

Figure 2.1: The contingency planning

As shown in figure (2.1), each plan developed must consider unpredictable events, globalization and competitors to generate the appropriate result.

2.3 Need for contingency planning

Managers need good plans to increase their profits and to optimize the use of their scarce resources. Recently, there appear new obstacles affecting plans, as, incomplete information, presence of exogenous and unpredictable events, accompanied by the uncertainty and complexity of the environment. All these events are the result of globalization and the growing number of competitors. In this respect, contingent plan are primordial to avoid the above mentioned problems, and to avoid harming expensive human and material resources.
However, classical plans consider only a set of actions that change the state of the world deterministically. This plans are blocked in many real situations. That's why, more robust plans are needed to consider the uncertain aspect of the environment.

2.4 The classification of contingency plans

Many research were developed to improve classical plans. So, many approaches are presented to catch environment uncertainty. In this context, there are three categories of planning under uncertainty: Conditional plan, Probabilistic plan, and Probabilistic and Conditional plan. In this section, we will present with more details and give some examples of these plans.

2.4.1 Conditional plans

The planning agent, during the execution time, can observe the state of the world in order to use conditional plans, i.e. generate plans that have the structure of branches (If-then Else). The fundamental question in conditional plans is how many contingencies that should be identified in order to avoid unforeseen events during the execution of the plan and to reach to the desirable goals [34]. Thus, we should enumerate and anticipate all the possible conditions. Among conditional plans we can cite CASSANDRA,

PlanPKS, CNLP.
CNLP plan is an extending of the SNLP plan (classical plan) that constructs conditional plans. CNLP is a branching plan that testing the conditions then deciding whether branch to tackle [6].

2.4.2 Probabilistic plans

In classical plans, the plan is considered as a sequence of deterministic actions which its execution contribute to a single outcome. The execution of an action a_i in the state s_i can lead to the deterministic state s_{i+1}, this transformation is called a transition function and denoted as follow: $f(a_i, s_i) = s_{i+1}$. However, in the probabilistic plans this function is replaced by a transition probabilities $P_{a_i}(s_{i+1}|s_i)$ [6] and the execution of an action can lead to several outcomes.

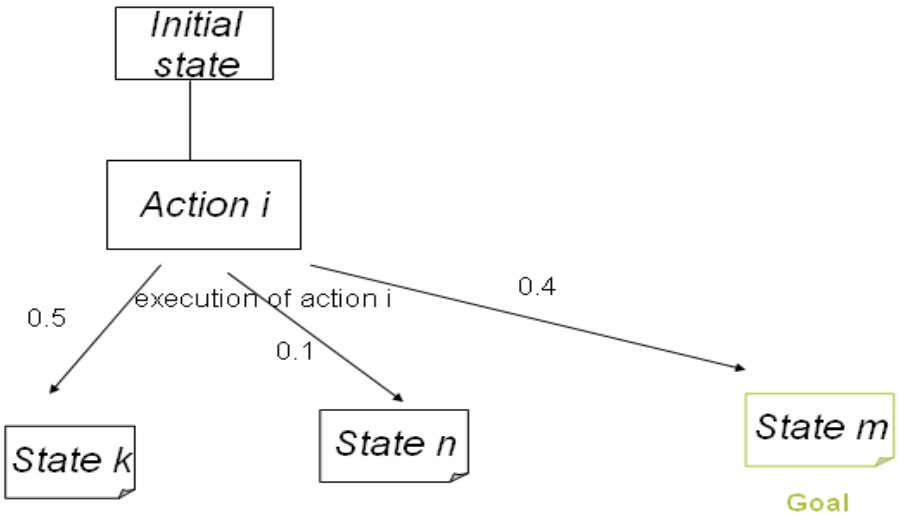

Figure 2.2: The probabilistic plan

The probabilistic plan can be viewed as a program that takes as input a set of states, a set of actions and an initial state and search for a goal state with higher probability of success. Thus, the crucial result of the probabilistic plan is to reach a goal state with a probability that exceeds a certain threshold. Among probabilistic plan we can cite BURIDAN, MAXPLAN.

2.4.3 Probabilistic and Conditional plans

This kind of plan can be considered as a combination of probabilistic and conditional plan where it generates conditional plans in stochastic domains. In other world, the observation states are coded as probabilistic outcomes then we construct the conditional plan.

2.5 An illustrative case

Consider the planning problem of moving computers and printers from the old laboratory (location A) to the new laboratory (location B). Our objective is to move the equipment unbroken to the new laboratory.
If we consider the classical plan, we consider only a set of state: {equipment in location A (the initial state), equipment unbroken in location B (the final state)} and a set of actions to be executed { move the equipment from A to B}. The planning process can be illustrated by the figure (2.3):

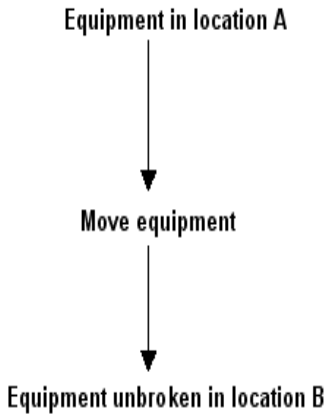

Figure 2.3: Classical plan

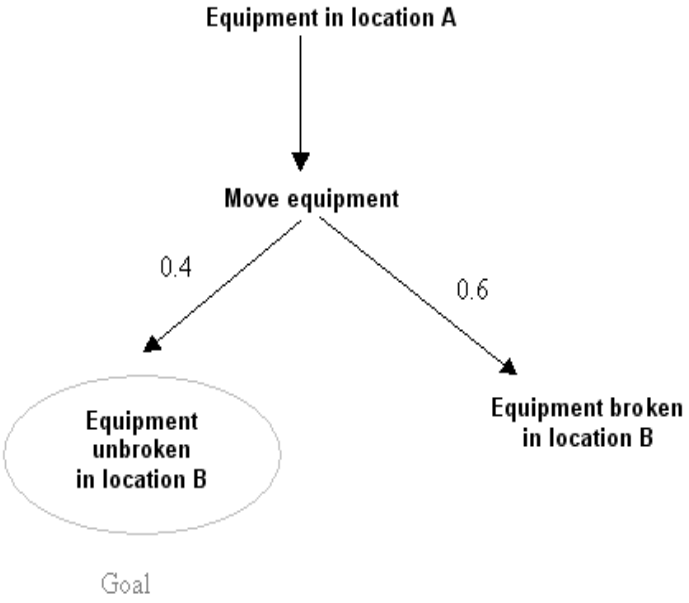

Figure 2.4: Stochastic plan

If we consider uncertain outcomes such as the weather conditions that could be assessed as rainy or sunny. In this case the deterministic or classical plan will be inefficient to reach the goal. That's why the situation can be modelized using probabilistic or stochastic plan as shown in figure (2.4).
An other approach can be used to reach the goal by applying conditional plan. As shown in figure (2.5), the conditional plan consists in testing the possible states, here we are considered just the state of weather, in order to take the appropriated action (as protect the equipment from the rain).

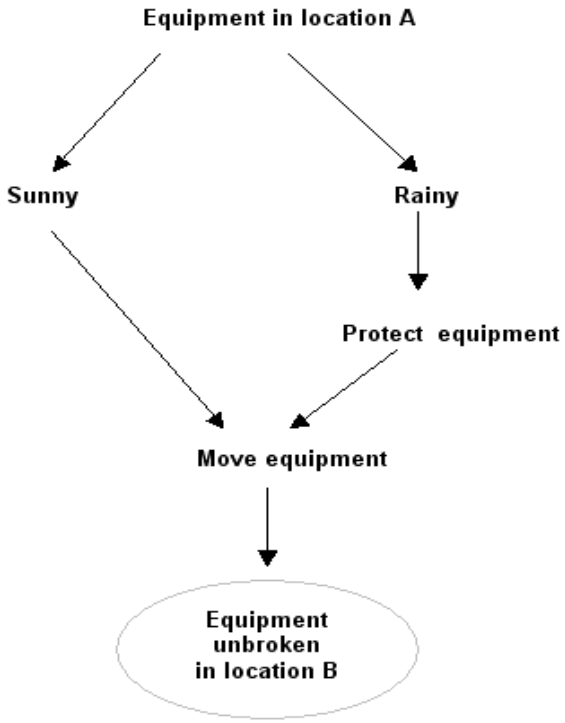

Figure 2.5: Conditional plan

2.6 Evaluation of contingency plans

In classical planning, the execution of the plan contributes to a single outcome. However, in contingency planning execution of the plan can lead to several results. Each result evaluated by the preferences of the project manager. In other words, the evaluation of a plan is done by the expected utility of it or the probability of success.

2.7 A survey of contingency plans

We need contingency to face random changes, disturbances and presence of uncertain events about the state of the environment. For that, the planning manager should be able to avoid outcomes and uncertainty. Throughout this section we propose an overview of contingency plans.

2.7.1 C-Shop plan

C-Shop (Conditional-Shop) is an extending of the classical plan Shop. C-Shop is a conditional plan that takes as an input a set of belief states, a list of tasks to achieve and a domain description and the output is a plan which is evaluated by a probability of success. C-Shop allows to code domain knowledge through procedures in order to provide a description how to solve the planning problem and to achieve goals. The main elements used in C-Shop are as follows [7]:

- Belief states: the uncertainty about the state of the environment is modeled by using belief states. Thus, we associate a set of observations for each belief state. This observations are used during plan execution in order to determine the condition to branch on.

- Methods: methods are used in C-Shop to control search and to provide the necessary information of how to generate and to decompose the tasks in primitive tasks.

- Operators: operators are used by the methods in order to achieve tasks with conditional effects and information obtained during execution. Thus, operators are applied to execute primitive tasks in a given belief states.

2.7.2 C-MaxPlan

C-MaxPlan is a conditional and probabilistic plan that generates conditional plan in probabilistic domains. C-MaxPlan transforms the planning problem into a stochastic satisfiabilistic problem. The objective of this plan is to find plan with higher probability of success [29].

2.7.3 Cassandra plan

Cassandra is a partial-order conditional plan. It proceeds by splitting the plan into a set of branches, one for each possible outcome. This plan uses explicit decisions steps that permit to the agent executing the plan to decide which plan branch to follow. Thus, Cassandra finds plans that permits to decide between courses of action that will succeed in different contingencies. The representation of the decisions steps in Cassandra provides to the agent plan information during execution of the plan [36]. Cassandra's representation of contingency plans based on three essential components:

- An action representation that supports uncertain outcomes;
- A plan;
- A system of labels for keeping track of which elements of the plan are intervened in which contingencies.

2.8 Conclusion

There is a growing need for contingent tools in combinatorial optimization problems, because they allow an efficient and feasible course of actions facing

the variation of the outside environment that can affect the objectives of the plan during the execution phase. Thus, generation of contingent plans is a difficult task. As a matter of fact, the process of developing a contingent plan is classified as NP-hard problem and in order to solve it in a polynomial time the researchers preferred the use of approximate approaches.
Hence, in the next chapter, we propose a survey of metaheuristics.

Chapter 3

Metaheuristics: An Overview

3.1 Introduction

Multi-objective optimization problems are characterized by several objectives functions that should be simultaneously optimized. These objectives are often conflicting. To solve this *NP*-hard combinatorial optimization problems, both exact and heuristic algorithms have been developed. The exact algorithms are essentially based on the Branch and Bound method. These algorithms are different one from another according to the way of generating the upper bound [30]. Due to their exponential time complexity, exact algorithms are limited to small size instances. Contrarily to this exact methods, the metaheuristics are powerful techniques used for solving a large number of *NP*-hard combinatorial optimization problems. For that, heuristic approaches can be beneficial when tackling complex combinatorial optimization problems. Hence, heuristics algorithms are designed to produce near-optimal solutions for large problems instance and it is the only way to solve problems quickly and efficiently. Several algorithms based on metaheuristics approach have been developed, for example: Algorithms based on Tabu Search (TS) approach, Genetic Algorithms (GA), Ant Algorithms (AA), Memetric Algorithms (MA), Particle Swarm Optimization

(PSO), Shuffled Frog Leaping Algorithms (SFLA). This evolutionary algorithms are inspired from different natural processes [14]. In this chapter, we describe an overview of Tabu Search, Genetic Algorithm and Ant System metaheuristics and we present their main principle and characteristics.

3.2 Tabu Search

The Tabu Search metaheuristic was first introduced by F.Glover in 1986 and its roots go back to 1970's [18]. This technique can be described as an iterative method based on a local search process [19]. Up-to-date, this method has shown a remarkable efficiency to solve a large kind of combinatorial optimization problems. Due to its flexibility, this technique can beat many *NP*-hard optimization problems. Combinatorial optimization problems were solved by Tabu Search , as, Krichen [27] applied the multiobjective Tabu Search algorithm for the multiobjective knapsack problem , Brucker et al. [3] propose a Tabu Search algorithms to solve the uniobjective RCPS problem.

3.2.1 TS: preliminaries

Tabu Search is considered as a local search optimization procedure which moves at each iteration to the best neighbor of the current solution. In order to avoid cycling back to solutions already visited, TS approach forbids moves to recently visited solutions by using a flexible memory in which its stored the history of the search process. TS allows the acceptation of solutions with inferior value of the objective function in order to escape from local optima. This approach based on many ingredients. The use of flexible memory can be viewed as the essential features of TS approach, beside the use of a neighborhood search method.

- The use of memory: TS uses the notion of memory to store some information related to the exploration of the solution space and the

attributes of more efficient solutions found. This information will be used to guide and to orient the neighborhood search procedure and to avoid the risk of cycling process.

- The neighborhood search procedure: The neighborhood search procedure is an iterative method based on the local search process. For each solution i we define a set of neighbors noted *N(i)*, and the move to the solution j is done by choosing the solution among *N(i)*, this choice is guided by information collected during the search process. This move will be stored in the tabu list to prevent cycling. So, the neighborhood search procedure is an improvement method. We can accept non-improving solutions, in order to escape from a local optimum.

- Intensification and diversification: The intensification favor the exploration of new solutions with good features. As mentioned before, the use of memory allows the search to be guided in a multiobjective environment. So, the good solutions are stored in order to examine their neighborhoods and to intensify the search procedure. The diversification encourage the search process to explore unvisited regions and to generate diverse solutions.

- The stopping criteria: The TS method is not a self stopping approach. Hence, with the function of the time searching, the criteria of stopping should be determined. This criteria can be fixed either through a certain number of iterations or through the non-amelioration of the good solution within certain number of iterations.

3.2.2 The algorithm

The outline Tabu Search algorithm are summarized as follows:

- Step 1: Generate an initial feasible solution;

- Step 2: Generate the set of the neighborhood of the current solution;

- Step 3: Choose the best neighbor;
- Step 4: Update the tabu list;
- Step 5: Update the set of efficient solutions;
- Step 6: Check if stopping criterion is met.

3.3 Genetic Algorithm

The Genetic Algorithms (GAs) were inspired by natural biological evolution. GAs were developed based on the Mendel's rules and the fundamental principle of Charles Darwin and have been developed and proposed by John Holland at the university of Michigan in 1970's [20]. Contrarily to TS, which is considered as sequential search method starts from an initial feasible solutions and attempt to improve it by moving to the neighbor of the solution, GA is a stochastic search method based on the mechanisms of natural selection and natural genetics in biological evolution [13]. This method work with a random population of solutions. GAs have been applied to a wide variety of combinatorial optimization problems and have been proved their efficiency. The first step of the optimization process using GA is to code the parameters of the objective functions. Then, solutions are represented as an array or a string, called *chromosome*. Each chromosome is evaluated and measured by a fitness functions. The improvement of the solutions at each generation is done by using the principle of the fittest process.

3.3.1 Adaptation of biological evolution

GA consists to encode a solution to a string called chromosome. Each chromosome composed of a set of elements called genes. Then, GA apply recombination operators to these solutions through crossover and mutation operators to produce offspring chromosomes. Thus, three natural operators are used for exploitation of the solution space: [14]

- Reproduction operators: consists of copying pairs of individual strings according to their fitness values. In the reproduction process we select the best individuals with higher fitness value in order to contribute in the next generation.

- Crossover operation: in order to maintain diversification and to generate differential individuals, crossover operator creates an offspring that contains a combination of genes from its parents. Thus, the crossover process consists to combine the best features of each chromosome by exchanging the genetic material between two individuals.

- Mutation operators: this operator used to diversify the solutions and to avoid stagnation around local optima. The mutation process consists to change position by modifying a 1 to a 0 and vice versa. Thus, this process contributes to produce new points in the solution space.

3.3.2 Difference and analogy between GAs and traditional methods

GAs differs from traditional methods in some ways. GAs work with a coding of the parameter of a specific problem, however, traditional methods work with the natural parameters themselves. To evaluate the solution quality GA requires only a fitness function. By contrast, traditional methods require auxiliary information like derivatives or local search procedures.
The analogy between GAs and traditional methods is summarized in the following table (3.1):

GA	**Traditional Methods**
Chromosome	Solution
Parameters are coded	Parameters are formulated
Fitness function	Objective function
Generation	Iteration
Using the fitness process	Using a local search process
Best-fit solution	Near-optimum solution
Population of solutions	Set of efficient solutions

Table 3.1: Analogy between GA and Traditional Methods

3.3.3 The algorithm

The parameter used in the GA are: population size, number of generation, crossover rate and mutation rate.

- Step 1: Generate a random population of N solutions or chromosomes;
- Step 2: Evaluate each chromosome by calculating the fitness function;
- Step 3: Select two parents or chromosome a and b according to their fitness function value;
- Step 4: Select an operator crossover or mutation;
  Generate an offspring c = crossover (a and b) or
  Generate an offspring c = mutation (a and b);
- Step 5: Evaluate the offspring c by the fitness function;
- Step 6: If c is better then replace the worst chromosome by c;
- Step 7: Check if stopping criterion is met.

3.4 Ant System

As the name suggests, the Ant System (AS) metaheuristic was built on the observation of the behavior of real ants while they are searching for food. Indeed, a colony of ants seems very easily able to find shortest paths between nests and food sources. The observation of this phenomenon allowed researchers to notice that, during their movements from nests and food source, ants deposit a chemical substance called pheromone on the path and other ants can smell this substance and thus choose the path with higher concentration of pheromone to go through. Over time, pheromone accumulate faster on the shorter path because ants using the shortest path will be back faster. This substance evaporates by time and makes less desirable path more difficult and less reinforced.

3.4.1 The Natural Archetype

It is easily observable that natural ants are collectively capable to find the shortest paths between their nest and a food source. Thus, while walking ants drop a substance called pheromone. Other ants are able to smell this leftover of their fellows and tend to choose a path with a higher pheromone concentration. Consequently pheromone trails are established and after some time all ants of the colony will walk on the these route due to the high pheromone accumulation.

This process is graphically illustrated in figure (3.1). When an obstacle is placed on the trail the ants don't have any idea where to go. Consequently they try to find a way around the obstacle and since they cannot determine which of the two routes is shorter, approximately, ants arrive at a decision point will choose path A or B with equal probability (0.5), thus, half of them will choose the shorter path and the other half will take the longer one (A_1). Ants will continue to drop pheromone on the two paths. Ants that have chosen the shorter path will arrive much faster to the other side of the obstacle, therefore the quantity of pheromone accumulate and increase more

faster on the shortest path than on the longer one and influencing other ants to follow this path (A_2). After a short period, all the ants of the colony will choose the shorter path because of the higher pheromone concentration (A_3). Thus, the whole colony was able to find the shortest path between the nest and the food source by the cooperation through the pheromone trail.

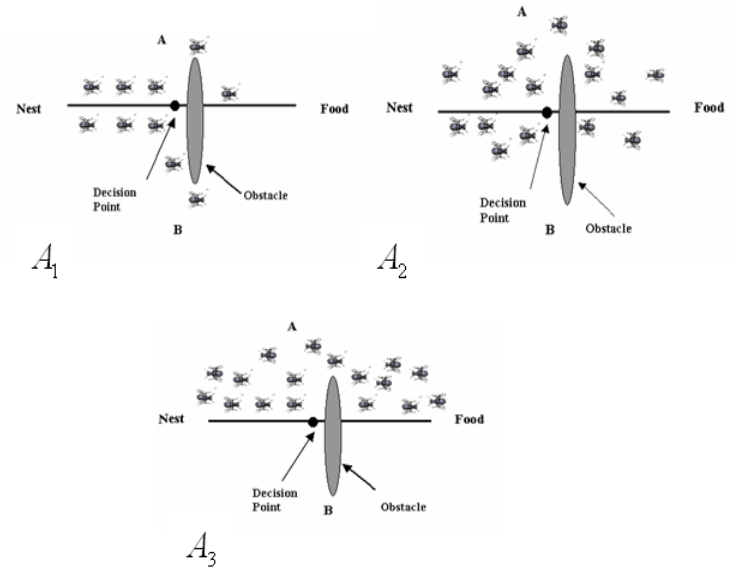

Figure 3.1: How real ants find the shortest path

3.4.2 Background

The AS is a multi-agent metaheuristic which has been successfully applied to several *NP*-hard combinatorial optimization problems. The Ant Colony Optimization (ACO) showed very good results since its first application by Dorigo in 1992 in his Ph.D thesis as a metaheuristic approach to solve the Traveling Salesman Problem (TSP), that's why, it was recently applied to various problems such as the Quadratic Assignment Problem, Vehicle Routing Problem [17], Graph Coloring Problem [9], Portfolio selection prob-

lem [11], etc.
Dorigo and Di Caro present the ACO metaheuristic and they defined a common model for discrete optimization problems and described the ACO metaheuristic as a colony of ants asynchronously moving through states of the problem by applying a stochastic decision policy. While ants are moving, partial solutions are being built and evaluated. The evaluation of partial solutions defines the update of pheromone trails to make. This pheromone information will direct the search of the future ants [12].

3.4.3 A brief description of the metaheuristic

The main motivation for using ACO metaheuristic by researchers is the adaptable character of the latter which makes of it a very useful tool to tackle a wide variety of combinatorial optimization problems, as well as the simplicity of the model on which the method is based. The real ant colony phenomenon described above inspired researchers to develop a new metaheuristic. This metaheuristic is especially suited to find solutions for difficult optimization problems. The ACO metaheuristic is composed of artificial ants that cooperate to find good solution.
In ACO metaheuristic, a colony of cooperating agents searches for a good solution to an optimization problem. Each agent or ant can already find a solution or at least a part of solution on its own but the optimal solution can not be reached individually. According to the problem considered, ants are given a starting state and they move through a sequence of neighbor states trying to find a shortest path. Moves are based on a stochastic decision policy directed by the ants'internal state, the pheromone trails and the environment information. Generally, the amount of pheromone deposited is proportional to the quality of a move an ant has made. Thus, better solution is obtained with more pheromone depositing. After an ant has found a solution, it dies. The main tenet of the ACO metaheuristic is the use of

a priori information concerning the problem data combined called heuristic information with a posteriori information concerning the structure of previous solutions obtained called pheromone trail. This combination is the stochastic decision policy called transition rule.

- The Pheromone trail: Cooperation and indirect communication between ants is the basic tenet to find good solutions quality to difficult optimization problems, and this is through the pheromone trail. It is considered as a numeric variable where each artificial ant of the colony can read or write information concerning the problem representation.

- The transition rule: It is applied to determine the next position of the ant and it is based on two parameters: the pheromone trail which represents the posteriori information and a heuristic information that represents the priori information. In This respect, each ant of the colony apply the transition rule in order to choose the next state to move to. Thus, the transition rule is a stochastic function which influences the process of selection the next state.

- Pheromone update: The pheromone update is essential and basic phenomenon to avoid stagnation situation, i.e. in which all ants build at each cycle the same solution, and to motivate the exploitation and exploration of the search space.

The analogy between Real Ants and Artificial Ants can be summarized in this table (3.2):

Artificial Ants	**Real Ants**
Searching the solution space	Searching their environment to find food
Equipped with local heuristic function to guide their search	Ants exchange information via pheromone
Use of a priori information	Randomly
Use of a posteriori information	Pheromone trail
Change a numeric information	Deposit a chemical substance called pheromone
Artificial ants die after finding a solution	Real ant still alive

Table 3.2: Analogy between real ants and artificial ants

3.4.4 The algorithm

The steps of the Ant Colony algorithm are summarized as follows:

- Step 1: Initialize the pheromone trails and parameters;
  Assigning Ants to the starting node;
- Step 2: For each Ant Do
  - Move to the next node by applying the transition rule;
  - Perform a locale update;
- Step 3: Perform a global update;
- Step 4: Check if stopping criterion is met.

3.5 Conclusion

We presented in this chapter an overview of three metaheuristic approaches namely: the Tabu Search approach based on a local search procedure for neighbors, the Genetic Algorithms inspired from the principle of the natural process of biological evolution and finally, the Ant System metaheuristic that consists in a constructive method based on the ants behaviors to find the

shortest route between their nest and a food source. These metaheuristics technique were extensively used by the researchers in multicriteria framework and represent efficient tools to tackle the multiobjective combinatorial optimization problems. In this respect, we will based on the Ant System metaheuristic to tackle the multi-objective RCPS problem. In the next chapter we will describe the problem and we will present the adaptation of this metaheuristic.

Chapter 4

An Ant based approach for multi-objective RCPS problem

4.1 Introduction

We study in this chapter a new RCPS problem characterized by:
- A multi-objective aspect.
- A contingency in the availability of resources.
- A variety of alternatives for tasks to be achieved: the multi-mode feature.

This problem can be formulated as a COA planning and it is classified as *NP*-hard problem [3] [5]. To solve this problem, it is suitable to apply meta-heuristics since it generates a good sample. In fact, the solution consists in assigning the available resources to each task in order to optimize the objective functions while satisfying a set of constraints. In this context, we propose to model the COA planning as a Multi-Objective RCPS (MORCPS) problem. Since the RCPS is a scheduling and resources allocation problem and which tries to find suitable resources allocation for a given COA schedule in order to optimize several objectives. To help the project manager

and to cope with the problem of planning with incomplete and uncertain information, our contribution consists in developing a new method based on the Ant System metaheuristic and multi-objectives concepts to handle uncertainty and to solve the MORCPS problem, we are interested here by the multi-mode RCPS problem [38] besides the aspect of contingency planning [6]. Hence, we define our problematic as follows: Given a set of limited and non-deterministic resources, what is the efficient, feasible and best COA planning/resources allocation according to several predefined goals?

4.2 Problem statements

The RCPS problem can be stated as a set of tasks, related by successor and predecessor constraints and where each task requires for its realization a various combinations of resources, to be assigned to a set of resources of limited capacity, where each resource can be used by various tasks. A resource can be human, material or financial and its availability is uncertain and is considered as a non-deterministic variable. In this respect, the mission is to allocate the necessary and available resources to the tasks over time optimizing several objectives and taking into account the evolutionary aspect of the environment besides the constraints of the problem. An Ant Algorithm is applied in order to construct an efficient and feasible COAs. The problem can be defined as:

- A set of tasks: where each task is characterized by a processing time d_i and can be executed by a combination of resources. The tasks are categorized as follows:
  - Strongly constrained tasks: are tasks with precedence and successor constraints and must respond to the time restriction as the maximum duration and the earliest starting time.
  - weekly constrained tasks: are tasks without precedence constraints but must respond to the time restriction.

- floating tasks: are tasks without precedence and successor constraints.

- A set of resources to allocate: due to the evolutive aspect of the environment the availability of resources is considered as a non-deterministic and it is handled according to a probabilistic variable P_j. Where P_j represent the probability of availability of resource j.

- A set of constraints to satisfy: to be feasible and realizable, each solution or COA must respect and satisfy the constraints of the problem such precedence constraints, resources availability,....

- A set of objectives to optimize: the performance of any approach proposed to solve a multi-objective combinatorial optimization problem is assessed based on their solutions'quality beside the computational execution time. Thus, the evaluation of the solutions'quality is according to the optimization of the objective functions.

The problem can be represented as an oriented graph, such graph allows the possibility to specify the combination of resources and the precedence constraint between tasks as follows in figure (4.1):

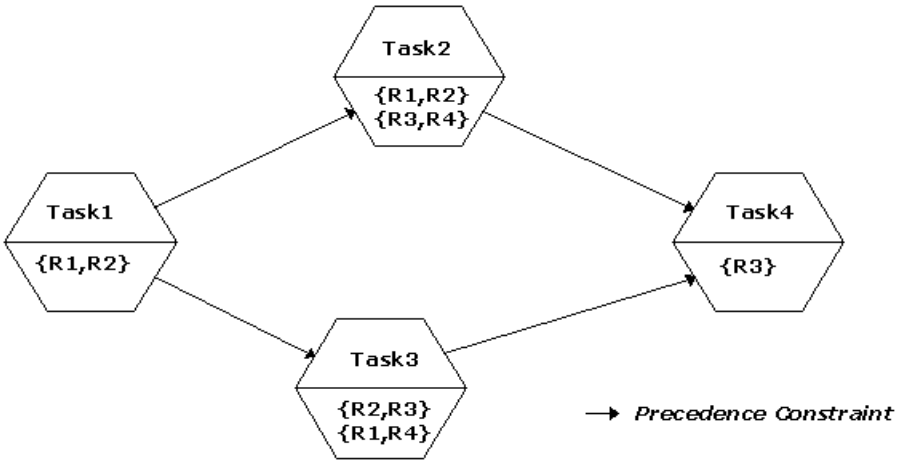

Figure 4.1: the structure of the RCPS problem

In this graph, the nodes identify tasks and their resources'combination and the edges represent the precedence relationship between them. Indeed, task 2 can be achieved by using the combination of resources R_1 and R_2 or the R_3 and R_4. This can only been achieved if task 1 has already been finished.

4.2.1 Notation

The notation and parameters used in our research are the following:

N	:	number of tasks.
t_i	:	the task i.
T	:	the set of tasks $\{t_i,\ i = 1,\ldots,N\}$.
k	:	number of resources.
M_i	:	number of modes that task i can be performed in.
$\|m\|$	:	number of resources of the mode m.
PR_i	:	the set of predecessors of task i.
PS_i	:	the set of successors of task i.
R_j	:	the available quantity of resource of type j.
r_j	:	the resource of type j.
q_{ijm}	:	the quantity of resource of type j required to task i being performed in mode m .
d_{im}	:	the duration of task i being performed in mode m .
s_i	:	starting time of task i.
P_j	:	the probability of availability of resource j.
l_i	:	finishing time of task i with $l_i = s_i + d_{im}$.
a_i^m	:	elementary action is the task t_i being realized by the mode m.
COA	:	the courses of action $\{a_i^m, i = 1,\ldots,N\}$.
CV_j	:	the in-use costs of resource of type j.
c_j	:	cost of resource j.
τ_{ij}	:	the amount of pheromone where task i is realized by the resource j.
τ	:	is the pheromone matrix.
η_{ij}	:	is a priori probability of availability of resource j used for task i.
C_{max}	:	the makespan $C_{max} = Max \sum_{i=1}^{N} l_i$.
P_{ij}	:	the probability that resource j successfully realize task i.
x_{ijm}	=	$\begin{cases} 1 & \text{if } t_i \text{ is realized by resource } j \text{ using mode } m. \\ 0 & \text{otherwise.} \end{cases}$
f	:	number of ants.
$\|PE\|$	:	number of potentially efficient solutions.

4.2.2 Mathematical formulation of the problem

A COA is a set of elementary actions a_i^m where each action is the realization of task i by the combination of resource m. The RCPS problem consists in scheduling a set of tasks with their resources in order to optimize a set of objectives and taking into account precedence constraints and resources availability. The evaluation of a given COA is according to the objective functions values.

Every task t_i has a processing time d_{im} and a time window $t \in [^{-}s_i, \underline{f_i}]$ representing the earliest starting time and latest ending time. Each task might begin if their predecessors $t_p \in PR_i$ are finished. More precisely, each task can have several predecessors and so it starts at s_i equal to the maximum of finish time of all its predecessors: $s_i = MAX(\sum_p f_p)$ where p represents the index of predecessors tasks of t_i.

Every resource r_j characterized by an availability calendar representing its available quantities over time, as shown in figure (4.2), and has specific characteristics such as: the probability of availability P_j and in-use cost CV_j. Thus, for each resource r_j assigned to task t_i, we have:

A cost: $c_{ij} = CV_j * q_{ijm}$.

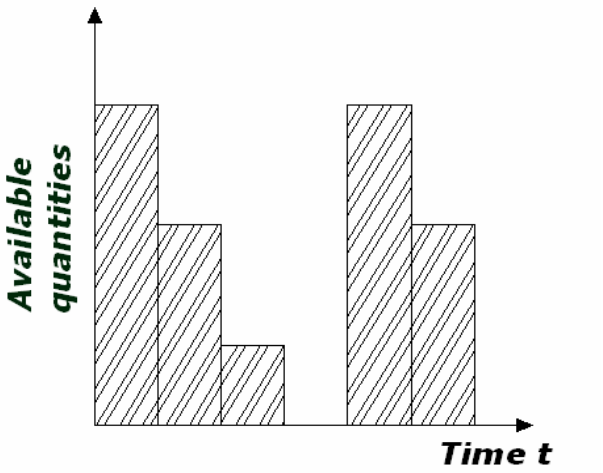

Figure 4.2: Availability calendar

The problem can be formulated as follows:

$$Minimize \;\; C_{max} = Max \sum_{i=1}^{N} l_i \tag{4.1}$$

$$Minimize \;\; \sum_{i=1}^{N} \sum_{m=1}^{M_i} \sum_{j=1}^{k} c_{ij} \; x_{ijm} \tag{4.2}$$

$$Maximize \;\; 1/N \sum_{i=1}^{N} (\sum_{m=1}^{M_i} \sum_{j=1}^{k} P_{ij} \; x_{ijm} / \sum_{m=1}^{M_i} \sum_{j=1}^{k} q_{ijm}) \tag{4.3}$$

Subject to:

$$\sum_{i=1}^{N} \sum_{m=1}^{M_i} x_{ijm} \; q_{ijm} \leq R_j, \quad j = 1,\ldots,k \tag{4.4}$$

$$s_i \geq Max_{p \in PR_i} (s_p + (\sum_{m=1}^{M_p} (\sum_{j=1}^{k} x_{pjm} / |m|) \; d_{pm})), \;\; i = 1,\ldots,N \tag{4.5}$$

$$\sum_{m=1}^{M_i} 1/|m| \sum_{j=1}^{k} x_{ijm} = 1, \quad i = 1,\ldots,N \tag{4.6}$$

$$x_{ijm} \in \{0,1\}, \quad i = 1,\ldots,N, \quad m = 1,\ldots,M_i, \quad j = 1,\ldots,k \tag{4.7}$$

The multi-objective RCPS problem consists in optimizing three objectives simultaneously:

- Minimize the makespan C_{max} equation (4.1);
- Minimize the COA cost equation (4.2);
- Maximize the probability of success of the COA equation (4.3).

Under a set of constraints denoted by equations (4.4 - 4.7); The resources availability constraint (4.4), means that each task i is performed by q_{ijm} quantity of resource j that can't exceed the available quantity R_j. Equation (4.5) is the predecessor constraint, guarantees that each task t_i starts if all

their predecessors $t_p \in PR_i$ have been finished. Constraint (4.6) ensures that exactly one mode is assigned to each task. Constraint (4.7) represents the decision variables, where:

$$x_{ijm} = \begin{cases} 1 & \text{if task } i \text{ is realized by resource } j \text{ using mode } m. \\ 0 & \text{otherwise.} \end{cases}$$

A solution to this problem can be viewed as sequences of synchronization tasks and resources, where each sequence corresponds to the realization of a given task by a combination of resources/mode. Thus, a sequence is an elementary action $a_i^m = (t_i, m)$.

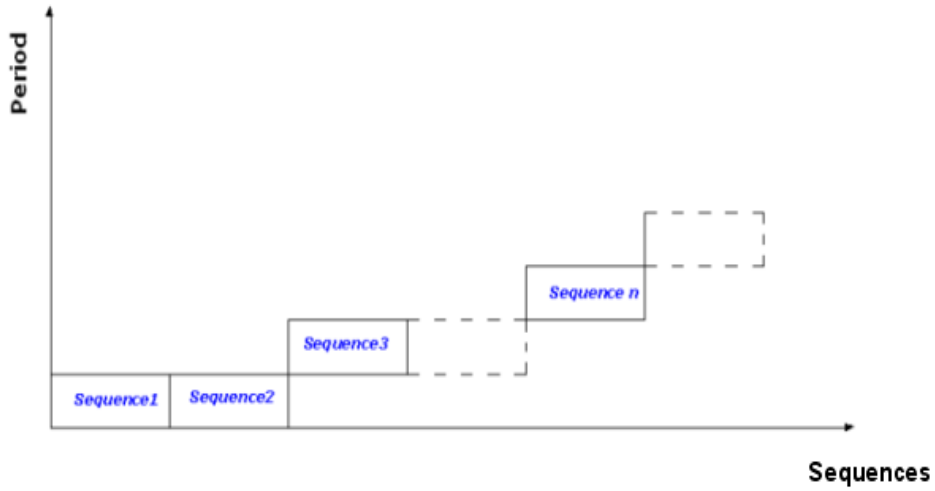

Figure 4.3: A solution structure to the RCPS problem

4.3 The Ant based approach

The AS is a multi-agent metaheuristic starts from null solution and builds at each cycle a feasible solution. It is inspired by the behavior of real ants when they forage for a food source and can be considered as a set of ants cooperating to find good solution. This metaheuristic is an iterative, stochastic and constructive approach and has provided an attractive compromise between the computational time and effort and the quality of the solution space. It was applied to a wide number of optimization problems, especially the single objective ones. In an attempt to find a sequence of realizable actions that allow an efficient utilization of scarce resources, optimizing several objective functions and predict unforeseen events, we will develop a new method based on the Ant System metaheuristics to optimize this problem. Thus, we have to solve an *NP*-hard multi-objective optimization problem, That's why, An Ant Algorithm is applied.
The main motivation for using this latter approach is its adaptable character which makes of it a very useful tool to tackle a wide variety of combinatorial optimization problems, as well as it is simplicity to be developed and in order to extend it for multi-objective problems and next there are few Ant algorithm developed to solve the RCPS problem. In this context, the elements of the Ant System approach will be adapted for use in our developed method.

4.3.1 The adaptation of the Ant System

For solving RCPS problem diverse method are proposed in literature, in this section we will propose an adaptation of the Ant System approach to our method as follows:

- **Pheromone trail**: pheromone trail has a high influence during the constructive phase, for that, The trail τ_{ij} corresponds to the quantity of trail deposit when task i is executed by resource j.

- **Heuristic information**: η_{ij} is considered as a priori information and it is computed by some heuristic function to indicate the desirably of moving from state i to state j. In our method, and due to the uncertain aspect of the availability of the resources, the heuristic information indicates the average of the probabilities of availability of resources j. Let us illustrate with this example: suppose that the quantity available of resource j is q_1 with probability p_1 and q_2 with probability p_2. Hence, the heuristic information is:
  $\eta_j = ((p_1 * q_1) + (p_2 * q_2))/(q_1 + q_2)$.

- **Transition rule**: is a stochastic search function influences and stimulates the ants'decisions during the construction process. This function is directed (i) by available pheromone trail τ_{ij}, (ii) by a heuristic function η_j and (iii) by a specific data of the problem.

4.3.2 The implementation strategy for the Multi-Objective RCPS problem

Our method starts with null solution and then each ant builds the solution in n iterations, where at each iteration one task is selected from the list of non-achieved tasks to be realized by a combination of resources. The choice of the resources is guided and directed by the problem characteristics and the available pheromone trail. And because we have to tackle a multi-objective combinatorial optimization problem we are defined two ACS colonies, the ACS-time colony and the ACS-cost colony, to allow each one the optimization of different objective functions. The goal of ACS-cost colony is to search the COAs with the minimum cost and maximum probability of success, and the goal of the ACS-time colony is to minimize the makespan of the project. In this respect, each colony has its own pheromone matrix and each ant of a given colony builds its own solution using only pheromone and heuristic information of its colony. The algorithm can be outlined in terms of the following features :

1. *Initialization phase*: initiate the pheromone matrix. The initial amount of the pheromone is initialized to small positive value τ_0. We have used two pheromone matrices:

- Pheromone matrix task-resource M_I: given n tasks and m resources

$$\tau_{ij} = \begin{cases} \tau_0 & \text{if resource } r_j \text{ contributes to the realization of task } t_i. \\ 0 & \text{otherwise.} \end{cases}$$

$$\tau_{ij} = \begin{pmatrix} \tau_0 & ... & 0 & \tau_0 \\ 0 & \tau_0 & ... & 0 \\ 0 & ... & 0 & \tau_0 \\ \tau_0 & 0 & ... & 0 \end{pmatrix}$$

- Pheromone matrix task-task M_{II}: given n tasks

$$\tau_{iu} = \begin{cases} \tau_0 & \text{if task } t_u \text{ is a floating task.} \\ 0 & \text{otherwise.} \end{cases}$$

$$\tau_{iu} = \begin{pmatrix} 0 & ... & \tau_0 & \tau_0 \\ \tau_0 & 0 & ... & 0 \\ 0 & ... & 0 & \tau_0 \\ \tau_0 & 0 & ... & 0 \end{pmatrix}$$

2. *Construction phase*: the two Ant colonies are activated simultaneously and uses independent pheromone trails:

- the ACS-cost: the goal is to minimize the cost function and to maximize the probability of success. Each ant of the colony use the pheromone matrix M_I and applies the transition rule P_{ij} to choose the adequate combination of the resources j to be assigned to a given task i. The transition rule of assigning task i to resource j is $P_{ij} = \tau_{ij} * \eta_j / \sum_{l \in \aleph_j} (\tau_{il} * \eta_l)$ where the $\aleph_j$ is neighborhood set of the combination of resources.

- the ACS-time: the goal is to minimize the total duration of the project (the makespan) C_{max}. Each ant choose randomly the combination of resources to be assigned to a task t_i and simultaneously use the pheromone

matrix M_{II} and applies the transition rule P_{iu} to obtain a neighbor by the selection of floating task t_u to be realized at the same period that t_i. Where $P_{iu} = \tau_{iu}/(d_u * \sum_{v \in \aleph_i} \tau_{iv})$ with $\aleph_i$ is the set of neighborhood of tasks i and d_u the processing time of the floating task u.

3. *Updating phase*: the two pheromone matrices are updated locally and globally.
- local update: during the construction phase and after building a solution, the pheromone trail intensity decreases over time to avoid convergence of the algorithm to local optimum and favoring the exploration of not visited areas of the search space. $\tau_{ij} = (1 - \rho)\tau_{ij}$, where ρ is a parameter determining the evaporation rate.
- global update: this procedure is applied at the end of the construction phase. Only the best solutions are allowed to deposit pheromone in order to generate new solutions in the neighborhood of these preferred ones and to favorite the diversification. $\tau_{ij} = \tau_{ij} + \rho(1/S^*)$, where S^* is the best value for respective matrix and objective.

4. *Filtering process*: this procedure is applied in order to select the potentially efficient solutions. In other words, we eliminate all the dominated solutions and we retain only the non dominated one.

The following chart summarizes the four phases of the Ant based algorithm:

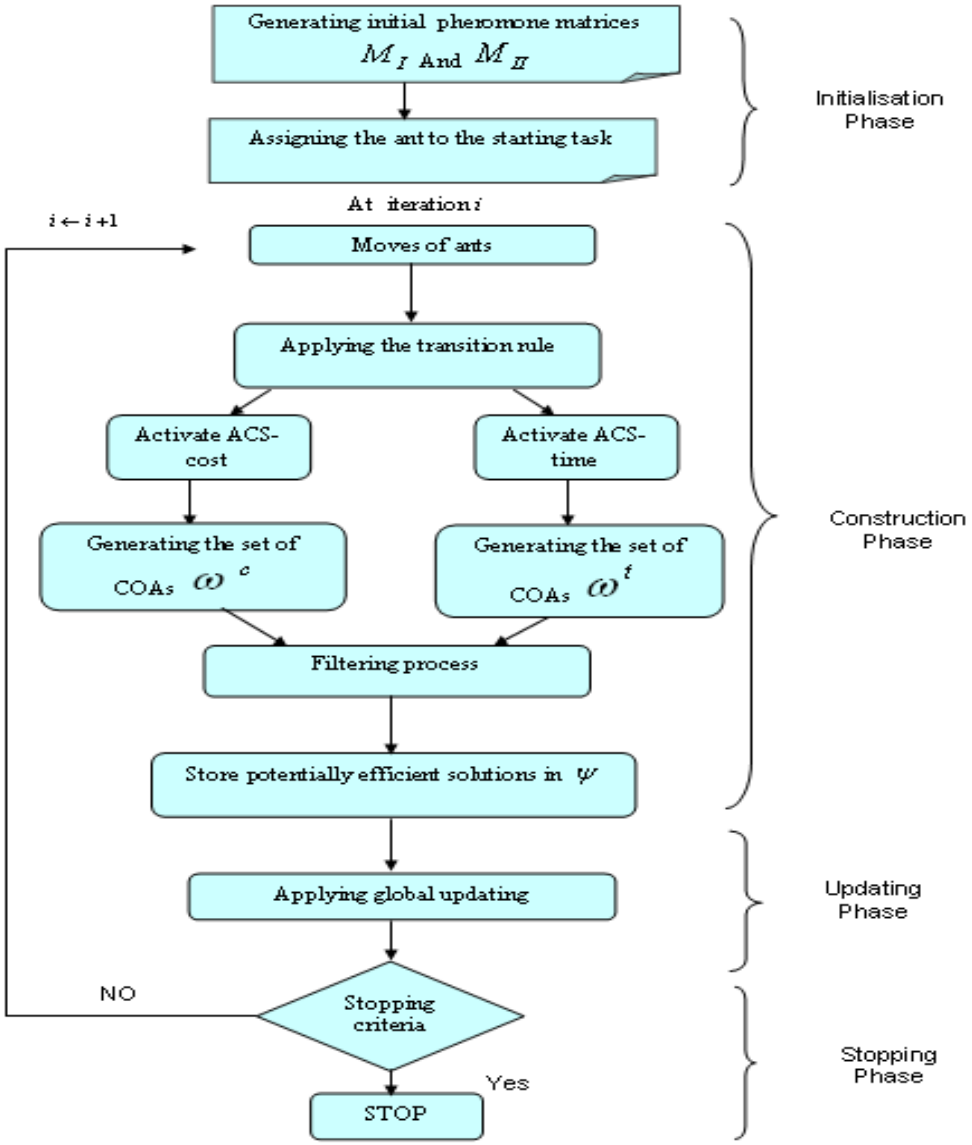

Figure 4.4: General scheme of the Ant based method

4.3.3 The global algorithm

- Initialization:
  - Initialize the pheromone matrix M_I
  - Initialize the pheromone matrix M_{II}
  - assign the ants to the starting task
  -Ψ: set of potentially efficient solutions($\Psi \leftarrow \emptyset$)
  -ψ^t: set of feasible solutions according to the ACS-time ($\psi^t \leftarrow \emptyset$)
  -ψ^c: set of feasible solutions according to the ACS-cost ($\psi^c \leftarrow \emptyset$)
  -f: number of ants of each colony

- Iterative Process:
  at iteration i
  Step 1:
  -for each ant f
  - Perform ACS-cost(f, M_I)
  - Perform ACS-time(f, M_{II})
  - $\psi^c \leftarrow \psi^c \cup \psi_i^c$
  - $\psi^t \leftarrow \psi^t \cup \psi_i^t$
  end for each
  - $\Psi \leftarrow \psi^t \cup \psi^c$
  Step 2:
  - perform global updating
  Step 3:
  - filtrate Ψ in order to have only the set of potentially efficient solutions
  Step 4:
  - If stopping criterion is met, stop
  - Else $i \leftarrow i + 1$ go to step 1

4.3.4 The ACS-cost

The ACS-cost is concerned to optimize the use of scarce resources.

- Initialization:
  - Initialize pheromone matrix task-resource M_I
  - Assign the ants to the starting task
  - A is the set of waiting tasks ($A \leftarrow \emptyset$)
  - T the set of tasks

- Iterative process:
  At iteration i
  Step 1:
  - Choose task $t_j \in T$, $T \leftarrow T \backslash \{t_j\}$
  - Check the constraint of precedence and go to Step 2
  - $A \leftarrow t_j$

  Step 2:
  - Construct a solution by applying the transition rule
  - Store the generating sequences in ψ^c
  - Perform a local update

  Step 3:
  - Perform a global update
  - Step 4:
  - If $T = \emptyset$ and $A = \emptyset$ then stop
  - Else $i \leftarrow i + 1$ go to Step 1

4.3.5 The ACS-time

The ASC-time is used to minimize the makespan C_{max} by changing the position of the floating tasks.

- Initialization:
  - Initialize pheromone matrix task-resource M_{II}
  - Assign the ants to the starting task
  - A is the set of waiting tasks ($A \leftarrow \emptyset$)
  - T the set of tasks

- Iterative process:
  At iteration i
  Step 1:
  - Choose task $t_j \in T$, $T \leftarrow T \backslash \{t_j\}$
  - Check the constraint of precedence and go to Step 2
  - $A \leftarrow t_j$
  Step 2:
  - Construct a solution by applying the transition rule
  - Store the generating sequences in ψ^t
  - Perform a local update
  Step 3:
  - Perform a global update
  - Step 4:
  - If $T = \emptyset$ and $A = \emptyset$ then stop
  - Else $i \leftarrow i + 1$ go to Step 1

4.3.6 A description of soft and hard environments

The Ant Algorithm is implemented using C and the development environment is the Visual C++ 6.0. The input data are stored in the set of files. The access to these data was done by using the 'fopen()' procedure.
The computer code was executed on a Pentium 4 with 256 Mo of RAM and a CPU of 2.53 GHz.

4.4 Examples

4.4.1 Example 1 ($n = 6$)

In this section, we will illustrate the Ant Algorithm by applying an example. The inputs to the algorithm are represented in table (4.1), which corresponds to a set of tasks to be accomplished by a various combinations of resources. For example, task t_2 can be processed by three ways of resources combination: the first consists in Using 3 unities of R_1 combined with 2 unities of R_3, the second consists in Using 3 unities of R_2 combined with 2 unities of R_4 and the third consists in Using 5 unities of R_4.

Tasks	combinations of resources and quantities		
t_1	$R_1(2)$,$R_2(1)$	$R_4(4)$	-
t_2	$R_1(3)$,$R_3(2)$	$R_2(3)$,$R_4(2)$	$R_4(5)$
t_3	$R_1(2)$,$R_2(1)$	$R_3(3)$	-
t_4	$R_3(2)$,$R_4(2)$	$R_1(4)$	-
t_5	$R_2(3)$,$R_4(1)$	$R_3(2)$	-
t_6	$R_1(1)$,$R_2(1)$	$R_4(2)$	-

Table 4.1: A RCPS problem with $n = 6$ and $k = 4$

The above example can be modeled as an oriented graph shown in figure (4.5), illustrating successor and predecessor constraints, where nodes represent tasks and edges correspond to the precedence relationship between tasks.

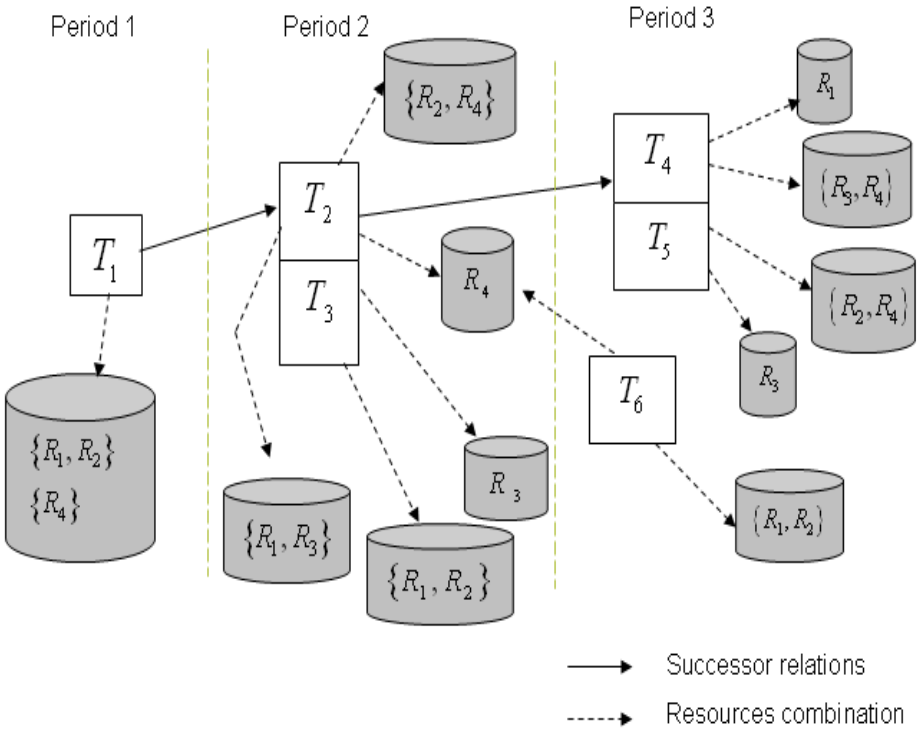

Figure 4.5: The structure of the RCPS problem $n = 6$ and $k = 4$

Applying the Ant Algorithm to the example in table (4.1), and varying the number of ants, we obtain the solutions enumerated below:

number of ants	**CPU time(s)**	**ACS-cost**	**ACS-time**
2	0	$\begin{pmatrix} 45 \\ 148 \\ 0.7 \end{pmatrix}$	$\begin{pmatrix} 30 \\ 150 \\ 0.66 \end{pmatrix}$
5	0	$\begin{pmatrix} 45 \\ 148 \\ 0.71 \end{pmatrix}$	$\begin{pmatrix} 30 \\ 150 \\ 0.65 \end{pmatrix}$
10	0	$\begin{pmatrix} 45 \\ 142 \\ 0.62 \end{pmatrix}$	$\begin{pmatrix} 30 \\ 148 \\ 0.72 \end{pmatrix}$
30	1	$\begin{pmatrix} 45 \\ 124 \\ 0.63 \end{pmatrix}$	$\begin{pmatrix} 30 \\ 150 \\ 0.72 \end{pmatrix}$
50	2	$\begin{pmatrix} 45 \\ 124 \\ 0.53 \end{pmatrix}$	$\begin{pmatrix} 30 \\ 150 \\ 0.72 \end{pmatrix}$
100	5	$\begin{pmatrix} 45 \\ 135 \\ 0.38 \end{pmatrix}$	$\begin{pmatrix} 30 \\ 150 \\ 0.66 \end{pmatrix}$

One of the solution obtained in the table $\begin{pmatrix} 30 \\ 150 \\ 0.66 \end{pmatrix}$ corresponds to ACS-time algorithm. The total duration of the project (the makespan):

C_{max}= max $\sum_{i=1}^{6} l_i$=30.

The cost of the utilization of the resources by the tasks:

$\sum_{i=1}^{6} \sum_{j=1}^{4} \sum_{m=1}^{M_i} c_{ij}\ x_{ijm} = 150$.

The probability of success:

$1/6 \sum_{i \in 6} (\sum_{m \in M_i} \sum_{j \in 4} P_{ij} x_{ijm} / \sum_{m \in M_i} \sum_{j \in 4} q_{ijm}) = 66\%$

4.4.2 Example 2 ($n = 11$)

We consider a RCPS problem with 11 tasks and 12 resources, $\begin{pmatrix} n = 11 \\ k = 12 \end{pmatrix}$, with the following data:

Tasks	Predecessors	combinations of resources and quantities			
t_1	-	$R_2(1)$,$R_3(2)$	$R_4(1)$,$R_6(2)$	$R_9(2)$	$R_{12}(1)$
t_2	t_1	$R_2(3)$,$R_6(2)$	$R_3(1)$,$R_5(1)$	$R_{11}(1)$	$R_{12}(2)$
t_3	t_2	$R_1(3)$,$R_3(1)$	$R_5(2)$,$R_7(3)$,$R_8(1)$	$R_{10}(3)$	$R_{11}(1)$
t_4	t_3	$R_1(1)$,$R_2(2)$	$R_4(1)$,$R_6(1)$,$R_8(1)$	$R_9(1)$	-
t_5	t_3	$R_1(1)$,$R_2(2)$	$R_4(3)$,$R_7(4)$	$R_{10}(2)$	-
t_6	t_1,t_5	$R_5(1)$,$R_6(3)$	$R_{11}(1)$	-	-
t_7	t_4,t_5,t_6	$R_1(3)$,$R_3(1)$	$R_9(3)$,$R_{11}(2)$	-	-
t_8	-	$R_2(1)$,$R_5(2)$	$R_{10}(1)$	-	-
t_9	-	$R_1(2)$,$R_3(2)$	$R_{11}(1)$	-	-
t_{10}	-	$R_2(2)$,$R_4(2)$	$R_6(2)$,$R_7(1)$	$R_9(1)$	$R_{12}(2)$
t_{11}	-	$R_3(2)$,$R_5(2)$	$R_{10}(3)$	$R_{12}(1)$	-

Table 4.2: A RCPS problem with $n = 11$ and $k = 12$

The solution produced by our method to the above example is represented as shown in the following table:

number of ants	**CPU time(s)**	**ACS-cost**	**ACS-time**
2	0	$\begin{pmatrix} 21 \\ 208 \\ 0.88 \end{pmatrix}$	$\begin{pmatrix} 17 \\ 169 \\ 0.95 \end{pmatrix}$
5	0	$\begin{pmatrix} 21 \\ 208 \\ 0.88 \end{pmatrix}$	$\begin{pmatrix} 18 \\ 167 \\ 0.64 \end{pmatrix}$
10	1	$\begin{pmatrix} 21 \\ 178 \\ 0.71 \end{pmatrix}$	$\begin{pmatrix} 18 \\ 136 \\ 0.51 \end{pmatrix}$
30	6	$\begin{pmatrix} 21 \\ 161 \\ 0.81 \end{pmatrix}$	$\begin{pmatrix} 17 \\ 146 \\ 0.82 \end{pmatrix}$
50	11	$\begin{pmatrix} 21 \\ 178 \\ 0.91 \end{pmatrix}$	$\begin{pmatrix} 18 \\ 185 \\ 0.66 \end{pmatrix}$

One of the solution is $\begin{pmatrix} 21 \\ 178 \\ 0.91 \end{pmatrix}$ it corresponds to ACS-cost algorithm. The total duration of the project (the makespan):

C_{max}= max $\sum_{i=1}^{11} l_i$=21.

The cost of the utilization of the resources by the tasks:

$\sum_{i=1}^{11} \sum_{j=1}^{12} \sum_{m=1}^{M_i} c_{ij}\ x_{ijm} = 178$.

The probability of success:

$1/11 \sum_{i \in 11} (\sum_{m \in M_i} \sum_{j \in 12} P_{ij} x_{ijm} / \sum_{m \in M_i} \sum_{j \in 12} q_{ijm}) = 91\%$.

In the following figure (4.6) we represent the variation of the number of efficient solutions in accordance with the number of ants. We noticed that the number of solution rises when the number of ants rises. For example, for 10 ants for each colony we have 11 non dominated solutions and for 30 ants we have 39 non dominated solutions.

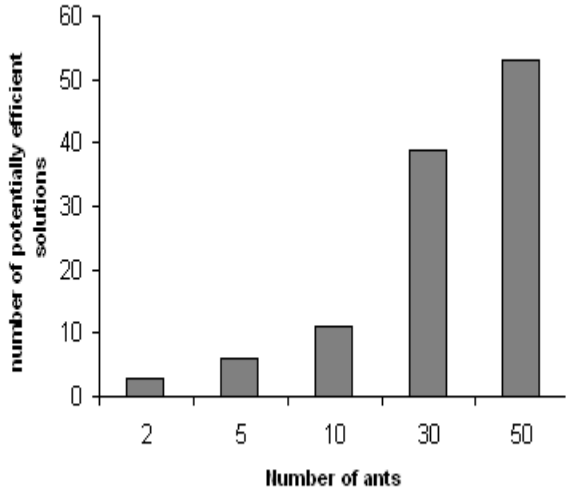

Figure 4.6: Chart of potentially efficient solutions corresponding to the number of ants $n = 11$

4.4.3 Example 3 ($n = 50$)

We consider a RCPS problem with $n = 50$ tasks. Then, we have to optimize the resources assignments in order to minimize the cost, the makespan and to maximize the probability of success. The cost and the probability of success depends on the chosen combination of resources.
Applying the Ant Algorithm to the proposed example and varying the number of resources we obtain the following table :

Tasks	Resources	number of ants	$\|PE\|$	CPU time(s)
50	5	10	2	4
50	10	10	6	4

We consider the problem with 50 tasks and 10 resources, $\binom{n=50}{k=10}$, applying our ant algorithm, using 50 ants, we obtain 11 non dominated solutions done in 20 second, which are represented in figure (4.7).

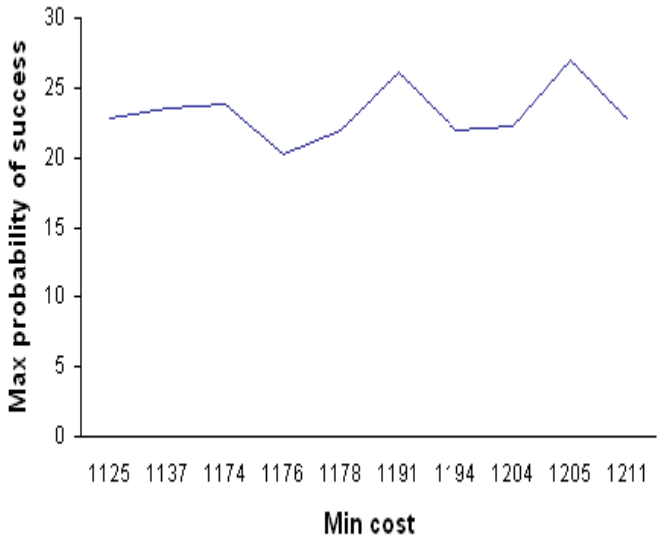

Figure 4.7: Chart of potentially efficient COAs $n = 50$

As shown in the figure (4.7), we can notice that if the cost decreases, the probability of success decreases, and vice versa. Thus, we notice that we can't minimize the cost and maximize the probability of success simultaneously. We can deduce that these two non commensurable objective functions

are conflictual.

4.5 Case studies

We enlarge the set of problems varying the number of tasks between 6 and 250. Since the studied problems are large, we adopt a random generation described as below:

- Initialization:
  - Initialize the number of tasks T
  - Initialize the number of resources K

- Iterative process:
  For each task i
  Step 1:
  - Assign the number of modes/combinations of resources M_i
  - Generate the set of successor S_i
  - Generate the set of predecessor PR_i

  Step 2:
  - Generate the set of resources
  - Construct the set of combination of resources.

In the following tables we report the numerical results provided by our Ant based method:

Problems	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
Number of tasks	6	6	6	6	11	30	50	50	50
Number of resources	3	5	4	5	12	4	3	5	3
Avg.number.predecessors	(1-2)	(1-2)	(1-2)	(2-3)	(1-3)	(1-2)	(1-2)	(1-2)	(2-3)
number of ants	5	5	10	10	10	10	10	10	5
CPU time(s)	0	0	0	1	1	4	5	5	2

Problems	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}
Number of tasks	50	60	100	100	100	100	150	200	250
Number of resources	10	4	3	3	5	15	15	20	15
Avg.number.predecessors	(1-2)	(1-2)	(1-2)	(2-3)	(2-3)	(1-2)	(1-2)	(1-2)	(1-2)
number of ants	10	10	10	5	10	10	10	10	10
CPU time(s)	6	10	19	4	8	8	35	60	59

4.5.1 A RCPSP with $n = 100$

Numerical experiments have shown that the obtained number of potentially efficient solutions depends on the number of resources used. Thus, when the number of resources rises the number of resources'combination rises. Hence, the number of diversified potentially efficient solutions rises also.

Tasks	Resources	number of ants	$\|PE\|$	CPU time(s)
100	5	10	4	8
100	15	10	6	9

The following chart corresponds to the variation of the number of potentially efficient solutions in accordance with the number of ants and this in order to maximize the number of diversified potentially efficient solutions. Indeed, for 100 ants we obtain 15 non dominated solutions done in 84 seconds and for 200 ants we obtain 18 non dominated solutions done in 164 seconds.

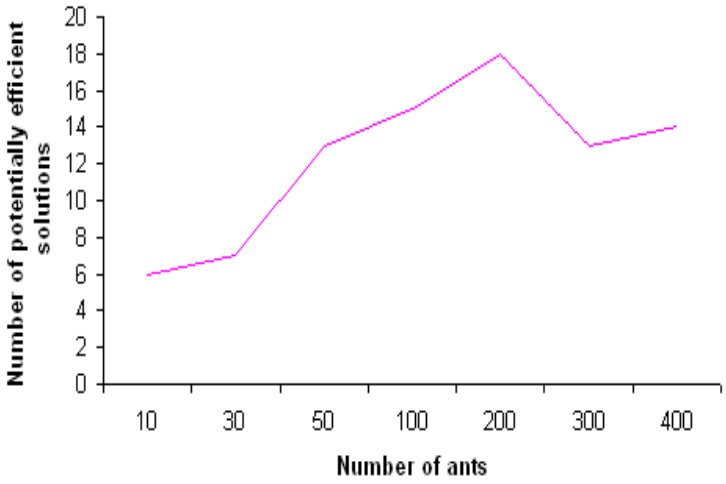

Figure 4.8: Chart of potentially efficient COAs $n = 100$

4.5.2 A RCPSP with $n = 200$

Let us considered a RCPSP with $n = 200$ tasks. When varying the number of resources we obtain the following table:

k	number of ants	$\|PE\|$	CPU time(s)
1	50	1	914
5	50	9	932
8	50	11	941
15	50	14	895
20	50	16	1070

Thus, the example of $\begin{pmatrix} n = 200 \\ k = 1 \end{pmatrix}$ is a classical RCPS problem where corresponds to the single mode RCPS problem. Each task has only one execution mode for that we have only one potentially efficient solution. However, the examples of $k = \{5, 8, 15, 20\}$ corresponds to the multi-mode RCPS problem where each task has several execution mode and where each mode has different value of the cost, the probability of success and the processing time C_{max}. For example, for $k = 5$ we have 9 potentially efficient solutions and for $k = 20$ we have 16 potentially efficient solutions and this due to the increased number of execution mode or resources'combination where the number of resources rises.

4.6 Lower Bound

Generally, for the *NP*-hard problem it is very difficult to find the optimal solutions. Thus, the solutions obtained from the heuristic methods represents a subset of the optimal ones. Hence, we employed the lower bound (LB) in order to frame the optimal solution of the multi-objective RCPS problem and to calculate the *Gap*: average deviations of the solutions generated by our algorithm from the lower bound value.

$$Gap = (Sol - LB)/LB \tag{4.8}$$

For that, we establish a lower bound on the makespan and the COA cost objectives denoted respectively by α_1 and α_2.
The lower bound of the makespan α_1 was selected to be the critical path length of the problem, which is equal to the technological earliest completion time [25] [35] and the lower bound of the cost α_2 was obtained by relaxing the resources availability constraint (4.9):

$$\sum_{i=1}^{N} \sum_{m=1}^{M_i} x_{ijm}\ q_{ijm} \leq R_j, \quad j = 1,\ldots,k \tag{4.9}$$

In the table (4.3) we drive a series of numerical examples for different size problems. As shown in table (4.3), for the problem P_3 the Gap_{α_2} is null,

Problems	α_1	α_2	Gap_{α_1}	Gap_{α_2}
P_3	8	124	0.12	0
P_5	26	68	0.23	0.13
P_6	56	702	0.08	0.13
P_{10}	44	982	0.27	0.05
P_{11}	115	1594	0.04	0.13
P_{15}	299	2630	0.1	0.14
P_{16}	289	4694	0.09	0.2
P_{17}	395	6357	0.08	0.17
P_{18}	519	6148	0.15	0.17
Avg. *Gap*			0.13	0.12

Table 4.3: Average deviations from the lower bound

so the solution coincides with the lower bound α_2. Hence, our algorithm generate the optimal solution for the COA cost.
The deviation from the LB for the two objectives is relatively small, for the objective α_1 the average Gap is about 0.13 and for α_2 the average Gap is about 0.12. These values are considerably interesting.

4.7 Computational experiments

In this section, we evaluate the performance of the set of potentially efficient solutions generated by our Ant algorithm denoted by AA. For that, we compare our results with the results generated by the tabu search algorithm

proposed in [5] denoted by *TS*. $|AA^*|$ and $|TS^*|$ are the number of potentially efficient solutions found respectively by the AA and TS methods.

Problems	P_3	P_5	P_6	P_{10}	P_{11}	P_{15}	P_{17}
Number of tasks	6	11	30	50	60	100	200
Number of resources	4	12	4	10	4	15	20
Number of ants	10	15	10	10	10	20	10
Tabu list size	4	7	9	20	10	25	25
$\|AA^*\|$	4	6	3	5	9	11	11
$\|TS^*\|$	3	3	4	6	3	9	5

Table 4.4: Comparison between the 2 methods TS and AA

The results summarized in table (4.4) indicate that our method provides more solutions than TS for large size problems P_{11}, P_{15} and P_{17}. However, TS finds more solutions for the problems P_6 and P_{10}.

CPU time(s)	P_3	P_5	P_6	P_{10}	P_{11}	P_{15}	P_{17}	P_{18}
AA	1	3	9	8	28	17	181	172
TS	2	4	24	13	40	47	212	111

Table 4.5: CPU time

Table (4.5) reports the CPU running times for problems $P_3, \ldots, P_{18}$ using AA and TA . We can see that the AA performs better than TS for the majority of the instances.

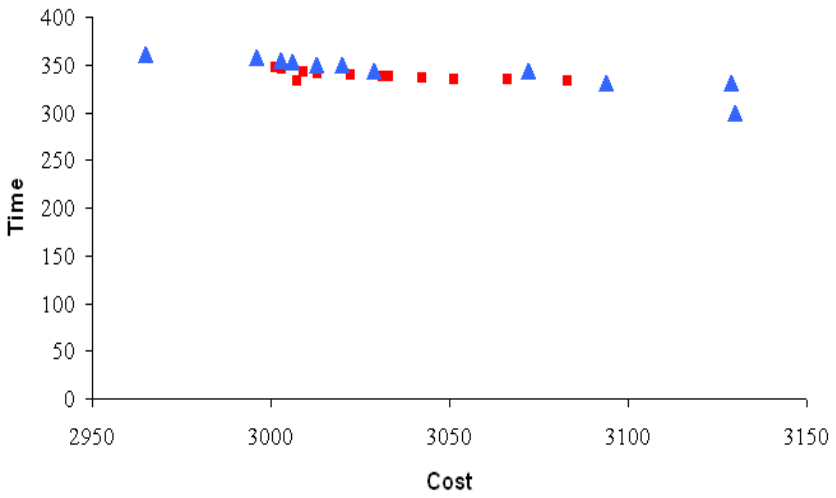

Figure 4.9: Comparison between TS(square points) and AA(triangular points) for problem P_{15}

We compare the quality of the results provided by the two methods as illustrated by figure (4.9). We can deduce that no common solutions were found by the two strategies. Thus, the pareto set generated by each approach is different from the other. We can notice from the figure (4.9) that the two strategies generate incomparable sets. For example the vector $\begin{pmatrix} time = 361 \\ Cost = 2965 \end{pmatrix}$ obtained by AA and $\begin{pmatrix} time = 334 \\ Cost = 3083 \end{pmatrix}$ generated by TS are incomparable. So, a better strategy consists in combining the AA with tabu search in order to get a better set of potentially efficient solutions.

4.8 Conclusion

We proposed in this chapter the RCPS problem with contingency planning and three objective functions: minimize the cost, maximize the probability of success and minimize the makespan. We proposed an Ant Colony based metaheuristic to get the set of potentially efficient solutions. We implemented our algorithm in C language and generated the solutions for problems varying from 6 to 250 tasks and 4 to 20 resources. Our method consists in finding a set of COAs optimizing the objective functions and satisfying a set of constraints. We defined two Ant colonies system to handle the multiplicity of the objective functions. The first colony is concerned to optimize the COA cost and the probability of success, for the second colony is developed to minimize the makespan. In order to approximate and to frame the optimal solution, we developed the lower bounds on the makespan and the COA cost objectives. The deviation from the lower bounds for the two objectives is relatively small.

General conclusion

The multi-objective Resource-Constrained Project Scheduling Problem is an *NP*-hard and complex problem due to its combinatorial aspect. In this research, we studied the multi-objective RCPS problem with contingency and we developed a new approach based on the Ant System metaheuristic in order to approximate the set of efficient solutions.

Our approach consists in generating two sets of artificial ants, some of them are about optimizing a single objective and other try to optimize multiple objectives. Each ant of the first colony moves in the sense to search a solution with minimum cost and maximum probability of success. However, the other ants of the second colony minimize the total duration of the project. These two colonies use independent pheromone trails. We adapted all the components of the Ant System metaheuristic as the transition rule, the pheromone trail and the heuristic information according to our problem settings. Experimental results show that the adaptation of this metaheuristic gives satisfactory results and the required time is relatively brief.

Finding an efficient solution for the multi-objective RCPSP is very difficult even impossible for large size instances. Consequently, we have calculated the lower bounds of the makespan and the cost objectives in order to frame the optimal solution. We noticed that our set of potentially efficient solutions is very coherent with these lower bounds. The average gap of the generated solutions is not far from the lower bound.

Bibliography

Bibliography

[1] Abbasi.B, Shadrokh.S, Arkat.J [2006]: "Bi-objective resource-constrained project scheduling with robustness and makespan criteria", to appear in Applied Mathematics and Computation.

[2] Al-Fawzan.M.A and Haouari.M [2005]: "A bi-objective model for robust resource-constrained project scheduling", International Journal of production economics Vol.96; pp.175-187.

[3] Baar.T, Brucker.P and Knust.S [1999]: "Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project-Scheduling problem". In: S.Voss, S.Martello, I.Osman and C.Roucailor (eds):Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers; pp.1-18.

[4] Belfares.L, Klibi.W, Nassirou.Lo and Guitouni.A [2007]: "Multi-objectives Tabu Search based Algorithm for Progressive Resource Allocation", European Journal of Operational Research Vol.177; No.3; pp.1779-1799.

[5] Ben Abdelaziz.F and Guitouni.A [2001]: "Méthode interactive pour la génération des suites d'actions", Technical Report, Institut Supréieur De Gestion, Tunisia.

[6] Blythe.J [1999]: "An overview of Planning Under Uncertainty", Management Science Vol.38; pp.1803-1818.

[7] Bouguerra.A and Karlsson.L [2002]: "Hierarchical Task Planning Under Uncertainty ", AI magazine; pp.37-54.

[8] Choi.J, Realff.M.J and Lee.J.H [2004]: "Dynamic programming in a heuristically confined state space: a stochastic resource-constrained project scheduling application", Computers and Chemical Engineering, Vol.28; pp.1039-1058.

[9] Costa.D and Hertz.A [1997]: "Ants Can Colour Graphs ", The Journal of the Operational Research Society, Vol.48, No.3; pp.295-305.

[10] Demeulemeester.E and Herroelen.W [1992]: "A branch and bound procedure for the multiple resource-constrained project-scheduling problems", Management Science Vol.38; pp.1803-1818.

[11] Doerner.K.F, Gutjahr.W.J, Hartl.R.F, Strauss.C and Stummer.C [2004]: "Pareto ant colony optimization with ILP preprocessing in multiobjective project portfolio selection ", to appear in European Journal of Operational Research.

[12] Dorigo.M and Di Caro.G [1999]: "Ant Algorithms for Discrete Optimization ", Artificial life, Vol.5, No.2; pp.137-172.

[13] Dowsland Kathryn.A [1996]: "Genetic Algorithms-A tool for OR? ", The Journal of the Operational Research Society, Vol.47, No.4; pp.550-561.

[14] Elbeltagi.E, Hegazy.T and Grierson.D [2005]: "Comparison among five evolutionary-based optimization algorithms ", Advanced Engineering Informatics; pp.1-11.

[15] Feng.C, Liu.L and Scott.A.B [1997]: "Using Genetic Algorithms to solve Construction Time-Cost Trade-Off Problems ", Journal of computing in civil engineering Vol.11, No.3; pp.184-189.

[16] Gagné.C, Gravel.M and Price.W.L [2001]: "A Look-ahead addition to the ant colony optimization metaheuristic and its application to an industrial scheduling problem ". 4th Metaheuristics International Conference .

[17] Gambardella.L.M, Taillard.E and Giovanni.A [1999]: "A Multiple Ant Colony System for Vehicule Routing Problems with time windows ". In : D.Corne, M.Dorigo and f.Glover (eds): New Ideas in Optimization, McGraw-Hill, London, UK; pp.63-76.

[18] Glover.F [1986]: "Future paths for integer programming and links to artificial intelligence", Computers and Operations Research, Vol.13; pp.533-549.

[19] Glover.F [1990]: "Tabu Search: A Tuturial", Interfaces 20:4 July-August; pp.74-94.

[20] Goldberg.D.E [1989]: "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company.

[21] Hajjem M'hirsi.S [2004]: "Planning Under Uncertainty : Contingent Plan ", Mémoire de DEA, Institut Supréieur De Gestion, Tunisia.

[22] Joshua Knowles.D [2001]: "Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization ", PhD thesis, Departement of computer Science, University of Reading UK.

[23] kolisch.R, Sprecher.A and Drexl.A [1995]: "Characterization and generation of a general class of Resource-Constrained Project Scheduling problems", Management Science, Vol.41, No.10; pp.1693-1703.

[24] kolisch.R and Hartmann.S [1999]: "Heuristic algorithms for the Resource-Constrained Project Scheduling problem: Classification and computational analysis", In: J.Weglarz (eds): Project Scheduling: Re-

cent Models, Algorithms and Applications, Kluwer Academic Publishers, Berlin; pp.147-178.

[25] kolisch.R and Hartmann.S [2000]: "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project Scheduling problem", European Journal of Operational Research Vol.127; pp.394-407.

[26] kolisch.R and Padman.R [2001]: "An Integrated Survey of Deterministic Project Scheduling", OMEGA International Journal of Management Science, Vol.29, No.3; pp.249-272.

[27] Krichen.S [1997]: "A Multiobjective Tabu Search Algorithm for Knapsack Problems with Multiple Objectives ", Mémoire de DEA, Institut Supréieur De Gestion, Tunisia.

[28] Landa Silva.J.D, Burke.B.K and Petrovic.S [2003]: "An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling". Technical Report, Automated Scheduling, Optimization and Planning Research Group School of Computer Science and IT, University of Nottingham, UK.

[29] Majercik.S.M and Littman.M.L [1999]: "Contingent Planning Under Uncertainty via Stochastic Satisfiability", To appear in the Proceedings of the Sixteenth National Conference on Artificial Intelligence.

[30] Martello.S and Toth.P [1990]: "Knapsack Problems: Algorithms and Computer Implementations", John Wiley and Sons.

[31] Matthew.B.W [1996]: "A Genetic Algorithm for Resource-Constrained Project Scheduling", PhD thesis, Department of Machanical Engineering, Massachusetts Institute of Technology.

[32] Merkle.D and Middendorf.M [2001]: "A New approach to solve permutation scheduling problems with Ant Colony Optimization", Appli-

cations of Evolutionary Computing: Procceedings of EvoWorkshops; pp.484-493.

[33] Merkle.D, Middendorf.M and Schmeck.H [2000]: "Ant Colony Optimization for Resource-Constrained Project Scheduling", Proceedings of the genetic and evolutionary computation conference (GECCO-2000); pp.893-900.

[34] Nifuler.O and Martha E.P [1996]: "Contingency Selection in Plan Generation", papers for the 1996 American Association for Artificial Intelligence.

[35] Patterson.J.H and Huber.W.D [1974]: "A Horizon-Varying, Zero-One Approach to Project Scheduling", Management Science Vol.20, No.6; pp.990-998.

[36] Pryor.L and Collins.G [1996]: "Planning for Contingencies: A Decision-based Approach", Journal of Artificial Intelligence Research; pp.287-339.

[37] Wei Feng.C, Liu.L and Scott Burns.A [1997]: "Using Genetic Algorithms to solve Construction time-cost Trade-Off problems", Journal of computing in civil engineering; pp.184-189.

[38] Yang.B, Geunes.J, William.J and OBrien [2001]: "Resource-Constrained Project Scheduling: Past Work and New Directions ", Technical Report, Departement of Industrial and systems Engineering, University of Florida.

Appendix A

An example with 100 tasks

Tasks	Predecessors	combinations of resources and quantities	
t_1	-	$R_1(3)$,$R_4(2)$	$R_3(5)$,$R_2(1)$
t_2	t_1	$R_3(2)$,$R_7(5)$	$R_2(2)$,$R_4(3)$
t_3	t_1	$R_1(2)$,$R_2(4)$	$R_6(3)$,$R_7(1)$
t_4	t_1	$R_5(5)$,$R_4(2)$	$R_3(1)$,$R_6(2)$
t_5	t_2,t_4	$R_9(5)$,$R_{10}(2)$	$R_8(6)$
t_6	t_3,t_4	$R_8(1)$,$R_5(3)$	$R_1(5)$,$R_9(2)$
t_7	t_2	$R_7(4)$,$R_8(3)$	$R_2(2)$,$R_3(2)$
t_8	t_5	$R_9(5)$,$R_{11}(2)$	$R_8(3)$,$R_{10}(3)$
t_9	t_6	$R_4(5)$,$R_5(5)$	$R_3(2)$,$R_8(1)$
t_{10}	t_6,t_7	$R_{11}(3)$,$R_1(7)$	$R_7(7)$
t_{11}	t_8	$R_5(4)$,$R_6(5)$	$R_2(2)$,$R_3(2)$
t_{12}	t_8,t_{10}	$R_5(3)$,$R_1(3)$	$R_4(5)$,$R_2(2)$
t_{13}	t_9	$R_7(2)$,$R_8(1)$	$R_6(3)$,$R_8(5)$
t_{14}	t_{10}	$R_3(1)$,$R_5(3)$	$R_1(6)$,$R_{10}(2)$
t_{15}	t_{12},t_{11}	$R_{11}(3)$,$R_{13}(2)$	$R_{14}(5)$
t_{16}	t_{13}	$R_{12}(3)$,$R_{13}(2)$	$R_{14}(2)$,$R_{16}(2)$
t_{17}	t_{12}	$R_3(2)$,$R_7(3)$	$R_{12}(5)$,$R_{14}(2)$
t_{18}	t_{14}	$R_4(5)$,$R_{11}(2)$	$R_8(6)$,$R_{13}(3)$
t_{19}	t_{15}	$R_{15}(2)$,$R_{16}(3)$	$R_2(10)$
t_{20}	t_{16}	$R_1(5)$,$R_2(3)$	$R_1$1(3),$R_{16}(5)$
t_{21}	t_{18},t_{17}	$R_3(4)$,$R_2(3)$	$R_{10}(2)$,$R_{15}(1)$
t_{22}	t_{17}	$R_8(5)$,$R_9(10)$	$R_3(4)$,$R_{10}(3)$
t_{23}	t_{20}	$R_1(3)$,$R_2(6)$	$R_4(3)$, $R_7(2)$
t_{24}	t_{19},t_{21}	$R_3(5)$,$R_4(1)$	$R_{10}(5)$, $R_{12}(5)$
t_{25}	t_{22}	$R_6(2)$,$R_8(5)$	$R_7(4)$,$R_1$0(2)
t_{26}	t_{25}	$R_4(5)$,$R_{11}(9)$	$R_1(2)$,$R_{10}(2)$
t_{27}	t_{23},t_{24}	$R_3(2)$,$R_4(3)$	$R_5(3)$,$R_6(3)$
t_{28}	t_{24}	$R_6(5)$,$R_7(2)$	$R_{10}(3)$,$R_{16}(4)$
t_{29}	t_{26},t_{27}	$R_3(5)$,$R_5(2)$	$R_{10}(11)$

Tasks	Predecessors	combinations of resources and quantities	
t_{30}	t_{27}	$R_2(1)$,$R_4(3)$	$R_3(2)$,$R_4(2)$
t_{31}	t_{28}	$R_5(2)$,$R_7(4)$	$R_4(5)$,$R_8(4)$
t_{32}	t_{28}	$R_1(3)$,$R_3(2)$	$R_{16}(6)$
t_{33}	t_{30}	$R_1(3)$,$R_5(3)$	$R_7(1)$,$R_8(3)$
t_{34}	t_{29},t_{32}	$R_6(5)$,$R_8(5)$	$R_3(2)$, $R_5(4)$
t_{35}	t_{33}	$R_7(4)$,$R_5(2)$	$R_{16}(5)$
t_{36}	t_{33}	$R_9(3)$,$R_{10}(4)$	$R_{16}(2)$
t_{37}	t_{34}	$R_7(5)$	$R_5(3)$
t_{38}	t_{35},t_{36}	$R_6(6)$,$R_{13}(4)$	$R_3(3)$,$R_5(4)$
t_{39}	t_{36}	$R_8(4)$,$R_{13}(2)$	$R_5(2)$,$R_6(2)$
t_{40}	t_{37}	$R_{11}(3)$	$R_3(3)$,$R_{16}(1)$
t_{41}	t_{38}	$R_{15}(3)$,$R_{16}(1)$	$R_1(2)$,$R_4(2)$
t_{42}	t_{38}	$R_{16}(5)$	$R_8(3)$,$R_{15}(5)$
t_{43}	t_{39}	$R_8(3)$,$R_9(2)$	$R_{10}(1)$,$R_{16}(3)$
t_{44}	t_{40}	$R_4(3)$,$R_5(3)$	$R_8(5)$
t_{45}	t_{42}	$R_3(2)$,$R_7(1)$	$R_6(3)$,$R_{14}(3)$
t_{46}	t_{42},t_{43}	$R_{14}(2)$,$R_{16}(1)$	$R_1(5)$,$R_3(2)$
t_{47}	t_{41},t_{44}	$R_{15}(4)$,$R_{16}(1)$	$R_1(5)$,$R_3(2)$
t_{48}	t_{45}	$R_7(1)$,$R_8(2)$	$R_{10}(3)$
t_{49}	t_{46},t_{47}	$R_8(1)$,$R_9(2)$	$R_6(3)$,$R_{10}(1)$
t_{50}	t_{48},t_{49}	$R_5(5)$,$R_6(3)$	$R_9(2)$
t_{51}	t_{49}	$R_{15}(3)$	$R_6(2)$,$R_8(3)$
t_{52}	t_{48}	$R_1(5)$,$R_5(3)$	$R_{15}(2)$,$R_{16}(7)$
t_{53}	t_{51}	$R_3(3)$,$R_4(4)$	$R_{20}(5)$
t_{54}	t_{50}	$R_{20}(3)$,$R_{19}(3)$	$R_3(1)$,$R_5(2)$
t_{55}	t_{52}	$R_{15}(5)$,$R_{18}(3)$	$R_{19}(2)$
t_{56}	t_{53}	$R_7(2)$,$R_{17}(2)$	$R_5(3)$,$R_{15}(3)$
t_{57}	t_{54},t_{55}	$R_{15}(2)$,$R_{20}(3)$	$R_1(2)$
t_{58}	t_{55}	$R_3(5)$,$R_4(5)$	$R_6(2)$,$R_7(2)$

Tasks	Predecessors	combinations of resources and quantities	
t_{59}	t_{54}	$R_{10}(2)$,$R_{17}(2)$	$R_{20}(2)$
t_{60}	t_{57}	$R_{14}(5)$,$R_{17}(3)$	$R_4(3)$,$R_5(3)$
t_{61}	t_{58}	$R_1(2)$,$R_4(2)$	$R_2(2)$,$R_5(3)$
t_{62}	t_{59}	$R_6(6)$	$R_{11}(2)$,$R_{12}(5)$
t_{63}	t_{60}	$R_{20}(4)$,$R_{19}(3)$	$R_{17}(5)$
t_{64}	t_{61}	$R_{17}(2)$,$R_{18}(2)$	$R_8(10)$
t_{65}	t_{61}	$R_5(2)$,$R_{19}(5)$	$R_7(5)$
t_{66}	t_{62}	$R_7(3)$,$R_8(2)$	$R_{13}(5)$,$R_{20}(4)$
t_{67}	t_{64}	$R_9(2)$,$R_{10}(3)$	$R_{11}(3)$,$R_{16}(5)$
t_{68}	t_{63}	$R_{16}(3)$	$R_{12}(5)$,$R_{14}(2)$
t_{69}	t_{65}	$R_{10}(2)$,$R_{18}(3)$	$R_{11}(2)$
t_{70}	t_{66}	$R_{20}(5)$	$R_{19}(5)$
t_{71}	t_{67}	$R_5(5)$,$R_7(8)$	$R_2(3)$
t_{72}	t_{68}	$R_1(5)$,$R_2(4)$	$R_5(20)$,$R_8(3)$
t_{73}	t_{69}	$R_4(4)$,$R_2(2)$	$R_{10}(6)$
t_{74}	t_{70}	$R_{20}(5)$	$R_{10}(7)$
t_{75}	t_{71}, t_{72}	$R_5(2)$,$R_6(7)$	$R_8(9)$
t_{76}	t_{73}	$R_2(3)$,$R_6(4)$	$R_{19}(10)$
t_{77}	t_{73}	$R_3(4)$,$R_8(3)$	$R_5(12)$
t_{78}	t_{74}	$R_1(1)$,$R_2(2)$	$R_{12}(5)$,$R_{14}(2)$
t_{79}	t_{75}	$R_4(3)$,$R_9(2)$	$R_{17}(3)$
t_{80}	t_{78}	$R_{11}(2)$,$R_{17}(1)$	$R_{20}(5)$
t_{81}	t_{77}, t_{76}	$R_{13}(3)$,$R_{20}(2)$	$R_1(10)$
t_{82}	t_{80}	$R_4(10)$	$R_5(2)$
t_{83}	t_{79}	$R_5(3)$,$R_7(2)$	$R_4(2)$,$R_{10}(5)$
t_{84}	t_{79}	$R_1(3)$,$R_4(2)$	$R_3(5)$,$R_2(1)$
t_{85}	t_{81}	$R_3(2)$,$R_7(5)$	$R_2(2)$,$R_4(3)$
t_{86}	t_{83}	$R_1(2)$,$R_2(4)$	$R_6(3)$,$R_7(1)$
t_{87}	t_{82}	$R_1(2)$,$R_2(4)$	$R_6(3)$,$R_7(1)$

Tasks	Predecessors	combinations of resources and quantities	
t_{88}	t_{84}	$R_9(5)$,$R_{10}(2)$	$R_8(6)$
t_{89}	t_{85}	$R_8(1)$,$R_5(3)$	$R_1(5)$,$R_9(2)$
t_{90}	t_{82}	$R_7(4)$,$R_8(3)$	$R_2(2)$,$R_3(2)$
t_{91}	t_{86}	$R_9(5)$,$R_{11}(2)$	$R_8(3)$,$R_10(3)$
t_{92}	t_{87}	$R_3(1)$,$R_5(3)$	$R_3(2)$,$R_8(1)$
t_{93}	t_{88}, t_{89}	$R_2(1)$,$R_4(6)$	$R_3(10)$,$R_7(2)$
t_{94}	t_{90}	$R_2(3)$,$R_3(3)$	$R_1(5)$,$R_4(5)$
t_{95}	t_{92}	$R_8(10)$,$R_9(2)$	$R_7(4)$,$R_5(3)$
t_{96}	t_{94}	$R_3(2)$,$R_8(2)$	$R_5(5)$,$R_6(3)$
t_{97}	t_{93}	$R_5(2)$,$R_6(2)$	$R_2(5)$,$R_3(1)$
t_{98}	t_{96}	$R_1(5)$,$R_2(5)$	$R_4(2)$,$R_5(4)$
t_{99}	t_{97}, t_{98}	$R_3(4)$,$R_6(3)$	$R_5(2)$,$R_7(2)$
t_{100}	t_{99}	$R_5(1)$,$R_6(3)$	$R_2(2)$,$R_7(4)$