SAPIENZA
UNIVERSITÀ DI ROMA

Methods to manage resource conflicts

Facoltà di Ingegneria dell'Informazione, informatica e statistica
Corso di laurea magistrale in Ingegneria Informatica

Giuseppe Prisco
Matricola 1895709

Professore
Alessandro Annarelli

A.A. 2022-2023

Abstract

The objective of this essay is to develop the topic of “Methods to manage resource conflicts”.

In the discipline of project management, one of the most important and relevant techniques in a project is the Time Management.
Usually the analysis of a project duration is performed through the use of the well-known Critical Path Method (CPM) in a scenario where resources are assumed to be unlimited. However, it is often the case that there is a natural constraint over the scarcity of resources that may delay the overall project duration.

The goal of this review is therefore to analyze this aspect of resource constraints by elaborating and studying some of the relevant works that have been written around this topic and to comment about the choices of specific approaches instead of others.

Starting from the analysis and presentation of the setting and the investigated problem, the review will explore some of the approaches employed by both Project Managers as well as Project Management Softwares to confront these augmented difficulties.
Additionally, the concept of slack (also called float) will be reworked in light of the changes brought by these techniques to better reflect the new methods employed in the resource constrained setting.

The review is composed of the following structure: first there is an introduction to the most common techniques employed in the literature followed by the description of the setting in which these techniques are employed, then a brief description of the investigated problem is presented, in which a clear identification of the inappropriateness of those methods is performed.
Subsequently, the study of the methods used in the new setting is performed by exploring both exact and heuristics approaches, followed by the presentation of the renewed concept of float.
Personal considerations about these topics are done during their presentation as well as gathered and expanded in a dedicated comments section at the end of the essay followed up by a conclusion to complete the work.

Index

1. Introduction

Time management in a project is a long time employed activity. One of the first techniques developed is the well-known Gantt Chart[1], a scheduling diagram used in project management to plan, track and coordinate specific activities in a project and showing a clear representation of the progress of the project itself. It is still widely used to this day and represents an essential tool among the project manager's resources. However, the Gantt Chart fails to consider the interdependence of the activities such as precedences between them and is generally difficult to manually update in case of many changes occurring in the schedule.

In order to overcome these limits other methods were developed to manage time in a project, for example reticular techniques.
Reticular techniques are based on a network diagram, composed of nodes and edges, to represent activities and their relationships.
Among the reticular techniques we recall the *Critical Path Method* (CPM), the *Project Evaluation Review Technique* (PERT) and the *Precedence Diagramming Method* (PDM). While there exist other reticular techniques like the Graphical Evaluation and Review Technique (GERT) and the Venture Evaluation and Review Technique (VERT), the first ones are directly cited in the papers analyzed and their authors base their considerations on them.

The *CPM* consists of listing all the activities of a project, for example starting from the Work Breakdown Structure[2] (WBS), and then identifying the dependencies between them. Then the creation of the network diagram is performed, assigning with the forward pass an early start (ES) and finish date (EF) to each activity and with the backward pass a late start (LS) and late finish (LF) date.
The computation of slack times (or float) is done to identify the flexibility of the activities of the diagram, that is the amount of time an activity can be delayed without affecting subsequent activities or project end date.

[1] "Gantt Charts" https://en.wikipedia.org/wiki/Gantt_chart

[2] "Work Breackdown Structure" https://en.wikipedia.org/wiki/Work_breakdown_structure

Generally, the following types of floats are computed: *total float*, *free float*, *independent float* and *interfering float*.

The *total float* is the amount of time that the finish date of an activity can be delayed without affecting the completion date of the entire project and is computed as the difference between LS and ES of an activity. Since the activity duration is fixed, it is also equal to the difference between LF and EF dates.

The *free float* is the amount of time that the finish date of an activity can be delayed without affecting the dates of any other activity in a project and is calculated as the difference between the earliest ES date among all the immediate successors of an activity and the activity's EF date.

The *independent float* represents the max float in the worst conditions, that is when all the predecessors of an activity finish at their LF date and all the successors start at their ES. It is computed as the difference between the earliest ES of an activity's successors and the latest LF of its predecessors, and the activity duration.

The *interfering float* is the maximum amount of time by which an activity can be delayed without delaying the completion date of the entire project but will cause the delay of the ES of some of its successors. It is calculated as the difference between total float and free float.

Once the various floats are determined, the critical path can be identified as the sequence of activities that must be finished on time in order for the entire project to be completed without delays. Any delay in critical activities will inevitably delay the rest of the project.

Differently from the CPM, the *PERT* technique incorporates uncertainty by making it possible to schedule a project without knowing precisely the durations of all the activities comprising it. The durations are therefore expressed in a probabilistic manner and we distinguish between the *optimistic duration*, *pessimistic duration*, *most likely duration* and *expected duration*.

The *optimistic duration* (d_{opt}) represents the minimum possible time required to complete an activity, assuming that everything proceeds better than expected.

The *pessimistic duration* (d_{pess}), analogous to the optimistic duration, represents the maximum possible time required for an activity to be completed.

The *most likely duration* (d_{ml}) represents the best estimate for the time required to complete an activity, assuming everything proceeds as normal.

The *expected duration* of an activity consists of the best estimate for the time required for an activity's completion, keeping in account that not always things proceed as normal. It is calculated as the sum of $d_{opt} + 4*d_{ml} + d_{pess}$, everything divided by 6.
Moreover the *variance* of an activity is calculated and it represents the variability for the completion time of the activity.
Finally, the probability to conclude the project at a time up to a specific time T is computed.

Similarly to the CPM, the *PDM* consists of identifying the activities of the project and their relationships and includes additional connection types apart from the *Finish-to-Start* (FS) connection present in the CPM.
These connections are: *Start-to-Finish*, *Start-to-Start* and *Finish-to-Finish*.
In the case of a *Start-to-Finish* (SF) edge, an activity B cannot cannot be completed until another activity A starts.
In a *Start-to-Start* (SS) connection an activity B cannot start until the activity A starts and lastly for the *Finish-to-Finish* (FF) arc we have that the activity B cannot be completed until A is completed.

I want to remind that the precedences between activities identified in the previous techniques are only "technological" dependencies and not "resource" dependencies and each of the previous methods assumes unlimited resource availability, thus they are not suited when also considering resource restrictions.

1.1 The General Setting

The setting employed to support this review has several characteristics, summarized in the following paragraph.

A project is composed of several activities also referred to as jobs, tasks or operations. All the data needed for activities is assumed to be integer value, available and deterministic.
Each activity is assumed to consume a defined set of resources which are classified according to type, value and category.

Based on the work proposed by Kolisch R. and Padman R. [1], the category classification includes *renewable resources* which are constrained on a period basis only, *nonrenewable resources* which are limited over the entire planning horizon, *doubly constrained resources* which are limited on both a period and on a planning basis and *partially renewable resources* with their utilization limited to a subset of the planning horizon.
The type classification is used to distinguish resources based on their function and the value represents their available amount.
Activities are often needed to be completed before other activities due to technological reasons and these relationships are depicted in the network diagram.

As suggested by [1], in the literature there are mainly two types of representations for the network diagrams belonging to reticular techniques: the *activity-on-arc* (AOA) and *activity-on-node* (AON).
In the *AOA* representation, the nodes of the diagram represent events and the arcs represent activities. Dummy activities are used to preserve the precedence relations and dummy nodes are used to capture the start and completion of the project.
In the *AON* representation, activities and their associated parameters (start and finish times, duration ecc.) are represented within the nodes and the precedence relations are represented by directed edges.

In contrast to Gantt Charts, both the AOA and AON representation graphically show the relationships between activities.
All the papers analyzed, including the work proposed by Kastor A. and Sirakoulis K. [2] which explicitly highlight it, depict the representation of a project as an AON network.

Moreover, the notion of a “large project” is given in the paper proposed by Jerome D. Wiest [3]. This type of project consists in an extensive, unique enterprise leading to a well-defined goal like the construction of a large plant or a major maintenance project.

1.2 Investigated problem

As already anticipated, techniques like the CPM, PERT or PDM implicitly assume unlimited availability of resources. Therefore they may produce unrealistic schedules due to the high limitation of common resources which are required by many activities in practice. This leads to a needed conflict resolution method whenever the concurrent demand for resources by different activities exceeds their maximum availability.
Thus the project manager (PM) needs a process to decide which activities should be scheduled first and which one must be delayed in order to achieve the minimum time increase to the overall project completion time while also ensuring a feasible resource allocation.
The general category of such problems can be referred to as *project scheduling problems* (PSP). Whenever it is the case that there is at least one category of constrained resources involved, we refer to the resulting PSP as a *resource-constrained project scheduling problem* (RCPSP).

Summarizing the work in [1], there are many objectives when speaking of project scheduling techniques. The ones cited are *makespan minimization*, *net present value maximization*, *quality maximization* and *cost minimization*.

Among the previous objectives, the focus of all the papers analyzed falls on *Makespan minimization*. The makespan is defined as the time span between the start and the end of the project and minimizing it reduces to minimize the maximum of the finish times of all the activities. It is easy to compare two different schedules for a given problem which differ only in the finish time of an activity and it can be said that the schedule which has the smaller makespan is at least as good as the other schedule.
Project scheduling problems with makespan objectives are generally represented by AON networks.

The *Net present value* (NPV) maximization is more appropriate when there are significant levels of cash flows in the project, in the form of start-up costs for activities and payments after completing parts of the project.

The goal of this criterion is to generate a cost-critical path and schedule of activities in contrast to the previous time-critical path and schedule objective.
Since these problems are characterized by a complex and combinatorial nature and are difficult to be represented in a mathematical form, optimal approaches to solve them have been successful only in small instances.

When maximizing the quality of the project is the most important objective and is explicitly expressed by project managers, we talk about *Quality maximization* and project scheduling is persecuted with this objective.

Lastly, *Cost minimization* can be divided into activity-cost and resource-cost minimization objectives. While in the activity-cost objective the way in which activities are performed results in direct costs to be minimized, in resource-cost objectives the schedule of activities indirectly influences the cost through the resources.

Taking into consideration the *Makespan minimization*, the RCPSP is formulated as in [2] and in the work proposed by Kolisch R. and Hartmann S. [4].
A project is composed of a set J = {0,1,…,n,n+1} of activities where 0 represents a dummy activity marking the start of the project and *n*+1 represents another dummy activity for the completion of the project. Each activity *j* has a fixed integer duration d_j, a start time s_j and a finish time f_j, for $1 \leq j \leq n$. Activities are characterized by precedence constraints for which an activity *j* cannot start before all its immediate predecessor activities $i \in P_j$ have been finished.
There exists a set K = {1, …, K} of renewable resources and each activity requires $r_{j,k}$ units of resource type $k \in K$ during every period of its duration d_j.
Each resource type *k* has a limited (constant) capacity R_k at any point in time.
Moreover all the previous parameters are assumed to be deterministic and for the dummy activities it holds that $d_j = 0$ and $r_{j,k} = 0$ for all $k \in K$.
Finally, let $A(t) = \{ j \in J \mid f_j - d_j \leq t \leq f_j \}$ be the set of activities that are being processed (active) at the time instant *t*.

The goal of the RCPSP is to find completion times for activities and a schedule for which both precedence and resource constraints are feasible and for which the makespan of the project is minimized.

The analogous conceptual decision model can be formulated as follows:

(1) $Min\ f_{n+1}$

(2) $f_i \leq f_j - d_j \qquad j = 1, \ldots, n+1\ ;\ i \in P_j$

(3) $\sum_{j \in A(t)} r_{j,k} \leq R_k \qquad k \in K\ ;\ t \geq 0$

(4) $f_j \geq 0 \qquad j = 1, \ldots, n+1$

The objective function, given by (1), minimizes the finish time of the unique dummy ending activity *n*+1 and thus the makespan of the project.
The equation (2) enforces the precedence constraints between activities while (3) indicates that at each time instant *t* and for each renewable resource type *k*, the resource requirements of active activities does not exceed the capacity of the resource.
Equation (4) is used to assign positive values to the finish times of each activity.

The paper written by Tormos P. and Lova A. [5] extends the precedence relationship to the ones described in the PDM. In particular we have the scheduled start time SST_j, the scheduled finish time SFT_j, for $1 \leq j \leq n$ as well as minimal time lags introduced for the generalized precedence relationships SS_{ij}, SF_{ij}, FS_{ij} and FF_{ij} where *i* represents a predecessor activity. The additional constraints added in the conceptual model are the following:

- $SST_j - SST_i \geq SS_{ij} \qquad (i,j) \in A_{SS}$
- $SFT_j - SST_i \geq SF_{ij} \qquad (i,j) \in A_{SF}$
- $SST_j - SFT_i \geq FS_{ij} \qquad (i,j) \in A_{FS}$
- $SFT_j - SFT_i \geq FF_{ij} \qquad (i,j) \in A_{SS}$

There exist mainly two types of approaches to solve the RCPSP just defined and we distinguish them between *exact approaches* and *heuristic approaches*.

Exact approaches aim at finding the optimal solution to the problem and thus require a lot of computational time, even for medium size projects.

Among these methods we highlight *zero-one programming*[3], *dynamic programming*[4] and *implicit enumeration with branch-and-bound*.

On the other hand heuristic approaches tend to find the solution to the problem in a very short time, but this solution may not be optimal (however it can be very close to the optimum, thus acceptable by the PM as a "good" solution). Several methods have been studied for the makespan minimization, including *priority-rule based scheduling*, *truncated branch-and-bound*, *disjunctive arc concepts* and *metaheuristics techniques*.
I will present in a later section some of the studies conducted on these techniques.

Regarding the NPV maximization problem, heuristics approaches can be divided into *optimization-guided*, *parameter-based* and *metaheuristic approaches*.
As already mentioned, the problem analyzed in this review is focused on makespan minimization and thus I will not go any further into the analysis of the NPV maximization problems.

There exist other related problems, as proposed in [1], such as Capital constrained problems, time/cost trade-off RCPSP, Multiple execution modes, Scheduling with min/max time lags and the Payment scheduling problem which however fall out of the scope of this essay and will not be treated.

In conclusion, since the RCPSP belongs to the class of *NP-hard* optimization problems and given its combinatorial nature and the inefficiency of *exact approaches* in solving practical size projects, *heuristics approaches* seem to be a better method to be employed for finding a solution to the problem.
Nevertheless we will start our discussion with exact approaches in the upcoming section and will later analyze the heuristic approaches.

[3] "zero-one programming" https://www.investopedia.com/terms/z/zero-one-integer-programming.asp

[4] "dynamic programming" https://www.geeksforgeeks.org/dynamic-programming/

2. Exact Approaches

This section aims at introducing the methods to find optimal solutions, which however are not time efficient, to later compare them with the superior heuristic approaches.
The paper I choose to study in order to present this topic is the one proposed by Patterson J. H. [6]. He compares three of the exact procedures which seemed promising at the time for solving the RCPSP.

Reminding what is written in the introduction of this essay, we have that activities cannot be interrupted once begun and resources are assumed to be available in constant amounts during each period. Moreover, activities require an integer (and constant) amount of resources during their duration.

Each of the following examined procedures is enumerative based and searches the set of possible solutions in such a way that not all possibilities need to be considered individually. They start by solving only part of the RCPSP through relaxing certain constraints or ignoring some restrictions. While proceeding, the number of constraints ignored becomes smaller and a tree of partial schedules is produced. The techniques differ in how candidate schedules are considered in generating the tree and how intermediate and inferior schedules are recognized and are discarded.

The methods studied in [6] are the *Bounded Enumeration Procedure*[5], the *Branch-and-Bound Procedure* and the *Implicit Enumeration Procedure*.
Each procedure was found to be generally superior on a specific set of problems and their main features are explored.

After summarizing their characteristics, a brief recap of their strengths and weaknesses is presented at the end of this section.

[5] described in “An Algorithm for Optimal Project Scheduling under Multiple Resource Constraints” https://pubsonline.informs.org/doi/epdf/10.1287/mnsc.17.12.B803

2.1 Bounded Enumeration Procedure

This procedure starts by dividing each of the originally identified activities into a set of unit duration activities (consequently the number of unit activities created is equal to the original activity's duration). The RCPSP is then transformed into a problem of finding the shortest path in a finite directed network, called A-network.
This network is produced by generating the nodes, each corresponding to a precedence feasible assignment for a subset of unit activities (for now resource constraints are ignored). Then edges are added to the network, each connecting both a precedence and resource feasible assignment for subsets of unit activities.
Dynamic programming is then used to identify the shortest path in the generated A-network of partial solutions, thus determining the minimal schedule length for the RCPSP.
Since the number of assignments can grow rapidly at each stage, a number of elimination criterias is employed based on both precedence and resource constraints.
Among these criterias we have a *latest performance time* (LPT) parameter for which partial solutions containing activities that would be scheduled past their LPT are discarded. Additionally, since each activity requires a certain amount of resources during its duration, partial solutions which correspond to unfeasible resource assignments are also discarded.

One disadvantage of this approach is that the number of precedence feasible subsets considered after the subset elimination criteria is generally large.
On the other hand we have that the splitting of the activities belonging to the original network into unit duration activities does not increase the overall computational time.

2.2 Branch-and-Bound Procedure

In the branch and bound procedure, nodes in the solution tree correspond to precedence and resource feasible assignments for a subset of the project's activities.
Therefore activities not yet scheduled at a given node have their constraints or restrictions ignored, until future levels in the tree.

During the execution of the algorithm, candidate problems are selected for further investigation based on a *decision vector*, that is, a set of attributes of the partial schedule. This vector consists of a series of tie-breaking rules used to select the next candidate problem to consider.

Until there are ties on the ordered attributes of the decision vector, the next attribute is considered, until there are no more ties. Once there is no tie on an attribute, the remaining attributes are ignored. In the investigated case, the decision vector is composed of four ordered components: largest of three different lower bounds, current partial schedule time, total resource idleness from previous decisions and total number of immediate followers that could be scheduled in the given partial solution.
The three lower bounds of the first attribute for the decision vector are a precedence based lower bound, a resource based lower bound and a critical sequence lower bound (considering both precedence and resource constraints).
When good partial schedules are recognized, the enumeration procedure can be performed aiming at finding better upper bounds and a complete schedule.
During the execution of the procedure, candidate problems are eliminated using lower bound pruning (based on previous lower bound definitions) or dominance pruning (when an activity can be left-shifted resulting in a schedule both time and resource feasible).

However, the selection of candidate problems based on the decision vector could lead to vast amounts of computer memory required.

2.3 Implicit Enumeration Procedure

The implicit enumeration procedure investigated uses integer variables to reduce the overall memory requirement. The technique consists of a systematic evaluation (enumeration) of all possible finish times for the activities of the project.
The procedure starts by computing the earliest possible completion time for the first activity of the project and then subsequent activities are assigned their earliest feasible completion time. In the moment the last activity has been assigned a finish time, an improved schedule is found. During the augmentation phase, if an activity cannot be

assigned a precedence or a resource feasible completion time under a specific upper bound then the procedure backtracks and considers the next lower numbered activity.
In the case this activity would not delay the duration of the partial schedule, an attempt is repeated to identify a feasible completion period for the original activity.
In the other case, the next lower numbered activity would be considered and backtracking proceeds as before.
Possible inferior schedules are eliminated from consideration by activity number rather than feasibility tests, thereby accelerating the backtracking step.
Optimality is thus achieved when backtracking reaches the first activity of the project.

The differences in the previously identified approaches is that in the implicit enumeration procedure the order in which candidate problems are considered is determined before the execution of the algorithm, with respect to the branch-and-bound procedure in which the order is computed during the execution of the algorithm.
Additionally, both the implicit enumeration and branch-and-bound procedures grant to a node of the solution tree a precedence and resource feasible assignment for subsets of the project's activities whereas the bounded enumeration procedure corresponds to a node only a precedence feasible assignment.

Summarizing the characteristics of the algorithms dedicated to finding optimal solutions just described, they generally tend to produce a large number of feasible subsets to be explored at each step, thus making them slow when applied in real world cases.
Moreover they often require additional memory to store a partial tree of solutions, therefore making them also space inefficient.

The reasons why heuristic methods were developed and are still studied and expanded is because they are by far much faster than exact approaches and cheap in both memory requirements as well as time completion.
The next section will discuss some of the techniques employed in this field and highlights their main strengths.

3. Heuristic Approaches

The previous approaches aimed at finding exact solutions to the RCPSP.
However in many practical cases, they often require a lot of computational time and thus make them a less preferable choice for the problem.
On the other hand, although heuristic approaches may not always result in an optimum solution, they generally have the advantage over analytic techniques of arriving at an acceptable solution with much less computational effort.

In this section we discuss the basic approaches for the heuristic model as well as schedule generation schemes and different solution methodologies. The solutions presented in this essay are *priority-rule based heuristics* and *metaheuristics approaches*.

An heuristic represents a method of reducing search time in a problem solving situation.
One of the pioneers in the study of feasible schedules in the RCPSP is Wiest [3]. In his basic model, activities are selected from the list of those available at the moment and are ordered according to their total slack. In his work Wiest considers the total slack based only on technological constraints, however the concept of slack needs to be revised when resources consist of scarce amounts. In a later section we will investigate how the definition of total and free slacks is changed to reflect this situation.
The most critical activities will be scheduled first with a high probability, an element introduced to provide some randomness of activity assignments and the generation of different schedules on subsequent applications of the model. If an available activity is not scheduled in a period due to resource limitations, an attempt is made to schedule it during the following period. When an activity is postponed, it gains criticality and moves up in the priority list of activities.
In order to increase the utilization of available resources as well as to produce shorter schedules, additional heuristics have been proposed to enrich this basic model.

Each activity is associated with a normal amount of resources, a maximum amount of resources for crashing the activity and a minimum amount of resources required for

executing the activity. Furthermore, an activity is considered critical when its slack is less than some chosen value *k*.
If the activity to be scheduled is critical and sufficient resources are available, the activity is scheduled at its maximum amount of resources, therefore it is crashed.
If there are no sufficient resources to schedule the activity at maximum or normal resource amount, an attempt is made to obtain the remaining resources from other activities. This is done by searching for current active activities and borrowing from them the required amount of resources, if doing so does not delay the entire project completion time. Otherwise it may happen that an activity previously scheduled (which uses the same resources as the critical activity) has been postponed. In this case the remaining resources required by the critical activity could be taken by activities that could be rescheduled without affecting the project delay.
However, if all previous approaches fail and the activity cannot be scheduled even at the minimum amount of resources, a new attempt is made in the following period and the activity is delayed.
When considering non-critical activities, they are scheduled at a normal amount of resources. In case resources are insufficient even for their minimum amount, they are delayed to the next period.

In addition to trying to borrow resources or reschedule other activities, when there are critical activities not scheduled at their maximum amount of resources and there are some available, their amount of resources is augmented up to their maximum (this is done to speed up critical activities).
Finally, after all possible activities are scheduled for a given period, it may be the case that there are some leftover resources. A scan of all the active activities is done and they are ordered based on their total slack: the remaining resources are then assigned to activities in ascending order to increase as much as possible their amount of resources.

Paper [3] then introduces some aspects relating the quality of a schedule and possible means of their evaluation. I will compare them in a later section, introducing the “updated” concepts of floats.

Starting from the basic model just described, paper [4] further details heuristic approaches, describing *priority-rule based heuristics* and then *metaheuristic approaches*.

3.1 Priority-rule based Heuristics

A priority-rule based heuristics is composed of two components: the first one is the *schedule generation scheme* (SGS) while the second is the *priority rule* itself.
SGS are used to build a feasible schedule by the gradual extension of a partial schedule, that is an incomplete schedule over all the activities of the project.
There exist two different SGS which differ in the way the schedule incrementation is performed.
Serial SGS perform activity-incrementation while *Parallel SGS* perform time-incrementation. In the following subsections they will be briefly described.

3.1.2 Serial Schedule Generation Schemes

The serial SGS consists of $g = 1, \ldots, n$ stages in each of which an activity is selected and scheduled at its earliest time considering both a precedence and resource feasible assignment. With each stage there are two disjoint sets associated to it: S_g containing all the activities that have been scheduled and D_g containing the activities eligible to be scheduled. Since generally there may be activities that cannot be scheduled at a given stage because not all of their predecessors have been scheduled, the union of these two sets is not equal to all of a project's activities.
Additionally, we have the set of finish times F_g and the remaining capacity of resource k at time t equal to $R_k(t)$.
The algorithm starts assigning the dummy start activity a completion time of 0 and puts it into the partial schedule. At the beginning of each step g, the decision set D_g, the set of finish times F_g and the remaining capacities $R_k(t)$ at the finish times $t \in F_g$ are calculated. Then one activity j is selected from D_g and the finish time of j is computed by first identifying the earliest precedence-feasible finish time EF_j and later the earliest, also resource-feasible, finish time F_j.

For the resource unconstrained case, this algorithm always generates optimal schedules. For the RCPSP the set of schedules produced is called “active schedules”, that are schedules in which there do not exist activities that can be started earlier without delaying some other activity. For the problem of makespan minimization, optimal solutions will reside in the set of these active schedules.

3.1.3 Parallel Schedule Generation Schemes

Differently from serial scheduling, the parallel scheduling performs time incrementation and there is a schedule time t_g at each stage *g*. The sets considered are the complete set C_g containing all the completed activities up to t_g and the active set A_g containing all active activities up to t_g. The eligible set D_g contains all the activities that can be started at t_g with both precedence and resource feasible times.
The algorithm starts by setting the start activity to both the active and complete sets.
Then, two phases are repeated at each stage: the first phase identifies the next schedule time t_g and computes the sets C_g, A_g, D_g and the available capacity $R_k(t_g)$ while the second phase schedules a subset of eligible activities to start at t_g.
Identically to the serial SGS, the parallel SGS always produces optimal schedules for the resource unconstrained case.
This algorithm produces non-delay schedules, that are schedules in which there aren’t activities that can be started earlier without delaying other activities, even if activity preemption is allowed. This is a subset of the previously identified active schedules, however it may not contain an optimal schedule in the makespan minimization for the RCPSP.

3.1.4 Priority Rules

Starting from the SGS, priority rules are used to select the activity *j* from the decision set D_g. Similarly to the decision vector of the Branch-and-Bound procedure analyzed before, the priority rule consists of a mapping which assigns each activity *j* in D_g a value *v*(j) and an objective used to select the activity with the minimum or the maximum value. As before, in case of ties one or several tie-breaking rules need to be employed.

Many priority rules have been proposed in the past, including *minimum latest finish time* (LFT), *minimum latest start time* (LST), *shorter processing time* (SPT), *minimum slack* (MSLK) and *worst case slack* (WCS).
Since priority rules employ different types of information to compute *v*(j), they can be classified between network, time and resource based rules and lower and upper bound rules. Lower bound rules (respectively upper bound rules) calculate for each activity a lower (respectively upper) bound of the objective function value.
Moreover a further distinction can be made between local or global rules. Local rules employ only information available from the activity under consideration while global rules make use of a wider range of information.
Additionally, static or dynamic rules are defined with respect to the fact if the value *v*(j) remains constant or changes during the stages of the SGS.

Priority rules and SGS are combined in order to obtain different priority rule based heuristics. If the heuristic produces a single schedule it is referred to as a *single pass method* while if it generates more than one schedule it is called *multi pass method*.
Single pass methods are the first type of heuristics employed in the literature and they generally consist of applying one SGS and one priority rule to find a feasible schedule.
On the other hand, *multi pass methods* include a variety of combinations of SGS and priority rules. Among the ones analyzed in paper [4], there are *multi priority rule methods*, *forward-backward scheduling methods* and *sampling methods*.
The *multi priority rules methods* use the SGS multiple times and each time they employ a different priority rule, generally a lower quality rule as the stages continue.
Forward-backward scheduling methods iteratively schedule the project's activities by alternating SGS to perform forward and backward scheduling. The backward scheduling is similar to the forward scheduling seen before but with the reversed precedence network where the dummy end activity becomes the dummy start activity and vice versa. We will see that paper [5] will adopt a similar approach for computing the total float as well as the work proposed by Raz T. and Marshall B. [7].
Sampling methods are similar to single pass methods in the sense that they employ one SGS and one priority rule. However instead of performing the selection of an activity considering *v*(j), they bias the selection in a random fashion and a selection probability *p*(j) is computed instead. At each stage *g*, *p*(j) represents the probability that activity *j* is selected from D_g. Sampling methods differentiate based on how these probabilities are

calculated and we can identify random sampling, biased random sampling and regret based biased random sampling. Paper [4] further analyzes these differences but I will not go into any more details for simplicity reasons.

3.2 Metaheuristic Approaches

A more novel method is the one of using metaheuristic approaches. They have been used in other hard optimization problems and seem a good direction in being applied for the RCPSP.
All approaches encode the solution as a list with length equal to the number of a project's activities and this list is mapped into a schedule. Based on the algorithms used, several different representations for these lists have been adopted in the literature and I want to mention at least some of them in this review: we have the activity list representation in which the list corresponds to a precedence feasible activity list; the random key representation in which the list is composed of real-valued values assigned to each activity; the priority rule representation in which a list of priority rules is used; the shift vector representation in which the list is made up of shift vectors, each being a non-negative integer and finally the schedule scheme representation.

Among the metaheuristic approaches we distinguish between *simulated annealing*, *tabu search* and *genetic algorithms*.

In *simulated annealing*, a neighbor solution is generated from a starting solution by applying small perturbations to it. In the case the perturbed solution is better than the original one, the following search process will start from this new solution. In the other case the perturbed solution is only accepted with a probability that depends on two parameters: the magnitude of deterioration (or energy of the state) and on the temperature (a parameter that reduces over time to avoid accepting worst neighbors).

The *tabu search* instead evaluates every neighbor of a solution and moves to the best one found among them. In order to avoid coming back to a previously visited solution, this method employs a form of memory to keep track of worse states already visited.

Differently from the previous two strategies that employed a local criteria to select an activity in order to improve the schedule, *genetic algorithms* consider many different populations of solutions at once. Starting from a population, new solutions are produced by combining pairs of existing solutions (the process known as crossover) or by individually altering a solution (process known as mutation). When the new solutions are generated only the ones that fit the best will be part of the next population of solutions. The criteria to measure their “fitness” is usually based on the objective function value.

The first two proposed methods share a similarity in the sense that the simulated annealing can be seen as an extension of the greedy algorithm First Fit Search[6] while the tabu search extends the simple steepest descent search Best Fit Strategy[7].

In conclusion, heuristic approaches borrow some methodologies from exact approaches and change them by applying some set of heuristics, that are rules employed to diminish the searching space.
This results in reducing the search time in a general problem solving situation and fits in being used for the RCPSP, where the optimality of the problem is not always seeked but the time in finding the solution is much more important.

Another interesting topic is an activity’s criticality and network flexibility. Usually, project managers base their evaluations for these measures on the concepts of floats. However when resources limitations are considered, the float calculations in their basic form might produce misleading results.
Thus in the following section I will present how they are defined while keeping into account the resource constraints and how an activity’s flexibility can be derived from them.

[6] “First Fit Algorithm“ https://www.javatpoint.com/first-fit-algorithm-in-c

[7] “Best Fit Search Algorithm” https://www.geeksforgeeks.org/best-fit-allocation-in-operating-system/

4. New Float Definitions

As already mentioned in previous sections of the essay, the standard methods to compute the total float and free float are not suited to be used in a resource constrained environment. This is due to the fact that total float and free float, the two measures we focus on in this section, are computed using early and late dates calculated with techniques such as the CPM, PDM or PERT. However, we already explained that these techniques do not include any resource availability into consideration, thus generating indeed optimal schedules minimizing the overall project duration but they may not be feasible considering also resource requirements.

Paper [7] refers to the dates computed with algorithms applied to the RCPSP as scheduled dates. These dates differ from the dates obtained applying regular techniques since they are derived by also considering resource constraints.
The float calculation is changed to still preserve the original meaning of the concepts of total and free float while at the same time taking explicitly into consideration the impact of limited resource availability on the dates that are used to compute floats.

The types of dates introduced are the *early scheduled dates* and the *late scheduled dates*.
The *early scheduled start* date represents the earliest date that an activity can start on, provided that all of its predecessors have finished and the resources it requires are available. The *early scheduled finish* is simply obtained by adding to the early scheduled start the activity duration.
The *late scheduled finish* date is defined as the latest the activity can finish without delaying the project completion time and without exceeding resource availability restrictions. These represent the latest dates that satisfy both date constraints and resource availability constraints. The *late scheduled start* is then computed as the difference between the late scheduled finish and the activity duration.
However the algorithm used to obtain late scheduled dates will use different heuristics with respect to the computation of early scheduled dates.

The complete algorithm to compute early and late scheduled dates starts by first identifying the early scheduled dates in a forward pass using a standard resource allocation algorithm.
Then, starting from the completion date of the project just obtained, a backward pass is performed, scheduling activities as late as possible while ensuring precedence relationships between the activities.
When the activities are scheduled, a check is done to see if the aggregated resource requirements are satisfied with respect to resource availability.
In the case an overutilization of resources is detected, resource leveling heuristics are employed by scheduling earlier some of the designated activities requiring those resources.
As the authors suggest, the heuristics applied for the backward pass would somewhat be the opposite of those applied for the forward pass.
For example, a possible heuristic would be to first delay activities with the largest amount of total float and then resolve ties delaying first the activity with the latest scheduled start date. For the backward resource leveling heuristic, activities that could have been scheduled to finish the earliest would be moved earlier first.

After calculating the early and late scheduled dates, the *scheduled total float* can be defined as the difference between the late scheduled finish (respectively start) and the early scheduled finish (respectively start).
Similarly, the *scheduled free float* is defined as the amount of time an activity can be delayed beyond its early dates without affecting any of its immediate successors either by precedence dependencies or resource dependencies. The precedence constraint is reflected by the difference between the earliest early scheduled start among all of an activity's immediate successors and its early scheduled finish while the resource constraint is reflected by the number of periods in a row, following the early scheduled finish of the activity, in which there are sufficient resources available for the activity.
Thus the scheduled free float is the smallest between the previous two quantities.

When resources are not considered, it is immediate to see that the early (respectively late) scheduled start and finish become the early (respectively late) start and finish and the computation of the floats would be the same as, for instance, to the CPM.

In conclusion, paper [7] defines the scheduled floats involving the generation of two different schedules, however it is often possible to employ alternative resource allocations resulting in schedules with identical overall durations but with different timings for individual activities.
For example, an activity may be critical in one schedule but in another it may have significant float. Thus there may exist different “critical sequences” (a concept coined by the author of paper [3]) for a project, that are different sets of activities with zero float, with the same project completion time.

4.1 Alternative Schedules

The work proposed by Bowers J. [8] describes how to explore these multiple equivalent schedules.
Once a project is started, we may assume for simplicity that resource requirements are fixed and a single schedule is chosen. However, during the planning phase, it is advisable to consider the range of possible equivalent schedules since an activity's float can depend on both the resource allocation and the specific schedule adopted.
Although the equivalent schedules will have the same project completion time, some of those schedules may avoid placing in the critical sequence activities that are “problematic”. Thus all of these schedules should be considered when identifying the resource constrained floats.
In the case of simple networks it is possible to directly generate the equivalent schedules by inspection.
However in practical use cases, given the NP-hard nature of the RCPSP, heuristic approaches need to be employed.

The author suggests to apply some perturbation to the network in order to explore a wider range of alternative schedules.
The algorithm starts by computing resource constrained floats and latest start times and generating a first schedule $j = 0$ with duration D_j. For each activity i we have the early start ES(i, 0), the late start LS(i, 0) and the resource constrained float F_1(i, 0) in the first schedule.

Then the analysis for resource constraints was repeated adding the additional restriction LS(i, j) ≥ LS(i, 0) where *j* = *i*. This perturbation generates up to *n* schedules, where *n* is the number of activities. Although many of these schedules may be the same, there is the possibility to have distinct schedules with new critical sequences.
Activities are then subject to the additional constraint LS(i, j) ≥ LS(i, 0) + Δ , this time with *j* = *n* + *i*. This second perturbation generally produces schedules with a higher completion time but can also identify schedules with the same original duration for which there exists an activity with a greater LS, in turn revealing that there are other activities that took place earlier, often earlier than their ES(i, 0).
The network is then reversed with a similar approach of paper [7].
Activities are subsequently subject to the two constraints just described, in this reversed network.
The schedules found in the reversed network can be transformed and implemented for the original network, considering that the ES (respectively LS) of the original network is equal to D minus the late finish LF (respectively early finish EF) of the reversed network, where D is the project duration of the reversed network.

The effect obtained by reversing the network is to schedule the high priority activities as late as possible with the objective of still minimizing the project duration, thus scheduling other activities with a lower priority earlier.
Hence the analysis of the reversed network can produce equivalent schedules with different ES and LS for some activities.

Since each perturbation generates *n* schedules, the algorithm (which applies two perturbations in the forward pass and two for the backward pass) produces in total approximately $4 \cdot n$ schedules. We can therefore define the earliest early start EES of an activity as the minimum ES in the 4n schedules, the latest late finish LLF as the maximum between the LFs and the maximum float F_2 as the greatest between the F_1s.
Since most of the times the EES and LLS are obtained from different schedules and F_2 does not consider it, another measure for the float can be obtained as $F_3(i) = LLS(i) - EES(i)$.

Paper [7], which employed a forward and a backward pass for generating two different schedules, obtained a concept of float similar to F_3 described in [8]. The difference is that F_3 includes a wider range of alternative schedules.

In conclusion we have that F_1 relates to a particular schedule while F_2 and F_3 reflect the fact that in a resource constrained setting multiple equivalent schedules may exist.
F_2 identifies the maximum float of an activity across all the acceptable schedules with the same duration whereas F_3 measures the difference between the EES and LLS of an activity considering all the acceptable schedules.
Thus particular care should be taken when analyzing F_3 since the fact that there may be a wide interval between the ESS and LLS of an activity does not necessarily imply that it can start at any time between these two values.

Nevertheless both F_2 and F_3 can provide measures for the flexibility of a particular activity with a high value indicating that the precise timing of the activity is not important to the project.

4.2 Criticality of an activity

A more in-depth approach is the one proposed by Tormos P. and Lova A. [5] where, after finding the extended concepts of floats, they defined an activity criticality index based on them. They are in turn used to evaluate the flexibility of schedules and classify the activities.

The algorithm used to find the *Forward Total Slack* (FTS) and *Backward Total Slack* (BTS) is analogous to the one explained in paper [8]: starting from the project feasible schedule, activities are selected in decreasing order of their scheduled finish time (SFT) and the current Forward Free Slack (FFS) as well as the Latest Feasible Start Time (LST) and Latest Feasible Finish Time (LFT) for each activity are calculated under both precedence and resource constraints.

The FTS of activity *i* is obtained by repeating the previous step for each activity *j* such that $SST_j > SFT_i$. Of course, for the activities whose SFT is equal to the feasible completion time, they cannot be executed later.
Then, since the setting of the problem analyzed in paper [5] considers precedence relationships based on PDM, the FTS_i correspond to the minimum of LST/SFT minus SST/SFT minus SS/FF/SF/FS reflecting the same constraints listed for the PDM variation at the beginning of the essay.
Activities are then reversed from the order defined before and each activity *j* is scheduled in the earliest feasible position in the time window $[SST_j, \ldots, LFT_j]$ and its scheduled dates are updated.

The idea to compute the BTS of the activities remains basically the same as the FTS but with the ordering of activities as well as the scheduled start/finish times and late/early dates mirrored with respect to before.

We remind that the time windows defined by the FTS and the BTS can include both feasible and not feasible positions.

When considering the resource unconstrained case, activities can be easily classified as critical based on whether or not their slack is equal to zero but when resources are scarce and thus limited, a different concept for criticality needs to be established.
Depending on the values of the slacks previously obtained, paper [5] categorizes activities based on their criticality. This classification distinguishes between *Resource Critical activities*, that are activities critical only with respect to resources, and *Absolutely Critical activities*, those that are critical also with respect to precedence relationships.
More specifically, they can be classified as *Forward_Absolutely Critical Activity* (F_ACA), *Forward_Resource Critical Activity* (F_RCA) or *Forward_Non Critical Activity* (F_NCA).
When considering the BTS, the activities can be classified as *Backward_Absolutely Critical Activity* (B_ACA), *Backward_Resource Critical Activity* (B_RCA) or *Backward_Non Critical Activity* (B_NCA).

When an activity cannot be delayed while still maintaining both the feasibility of the schedule and the project completion time, thus having a FTS equal to zero, it is F_ACA.

In case an activity's FTS is greater than zero but it cannot be delayed in its FTS while maintaining the feasibility of the schedule, it is considered a F_RCA. If there are resources available or new resources show up, the project manager can resolve the unfeasible positions by delaying the scheduled dates of these activities.
In the last case in which an activity's FTS is greater than zero and has feasible positions in its FTS, then it is a F_NCA.
As for BTS, activities are classified similarly.

Having the activities classified, the authors defined a criticality index accounting for the total number of early feasible periods and later feasible periods in which an activity can be either started earlier or delayed further.
With this index the PM can more realistically reschedule the activities' dates in a resource constrained setting
The authors of paper [5] refer to this criticality index as the *Resource Constrained Activity Criticality Index* (RCACI). It is computed as the number of feasible periodes an activity can be anticipated in its BTS plus the number of feasible periods it can be delayed in its FTS.

It is easy to see that when resources are unconstrained and the activities are scheduled in their earliest dates, they will be either F_ACA or F_NCA and all of them will be B_ACA. Moreover the RCACI of each activity corresponds to the amount of its total float, thus the measures introduced in this study represent a generalization of the ones employed in the unconstrained setting.

The previously described approaches indeed consider resource constraints in their float definition, however they fail in identifying the hidden resource links when executing the backward pass.
The work presented by Kim K. and De La Garza JM. [9] addresses these problems and proposes a systematic algorithm to evaluate alternative schedules similar to the one described in [8]. Moreover, it also considers float calculations of parallel activities, again, not addressed in the previous papers, and refers to the difference between the theoretical remaining total float and the real remaining total float as the ‘‘phantom float’’.

The algorithm proposed in paper [9] is the *Resource Constrained Critical Path Method* (RCPM) and it consists of the following five steps:

1. At the start of the algorithm, the standard CPM is applied to find the early and late dates and the activities' total float. Notice how in this step resources are considered as unlimited.
2. Now a resource constrained scheduling technique is applied by employing the latest start time (LST) priority rule and ties are resolved by checking the shortest duration, total float and in the end the activities' ID.

   Until there are available resources to start an activity at its earliest start time, it is scheduled and the next activity is considered. When resources are not sufficient to fulfill the current activity's requirements, it is delayed until the completion of another activity whose amount of resources freed is enough for that activity to be executed for its whole duration.

   Thus, one or more resource links can be created as logical relationships between the current activity and any other activity whose resource requirements directly affect its start time, meaning that any delay of the predecessors will inevitably delay the start time of the current activity.

   In this step, only the EST and EFT of activities are identified.
3. To define the LST and LFT of the activities a backward pass is executed considering both the precedence links and resource links.

   Until now, the steps performed by the algorithm are pretty much the same as the previous analyzed approaches.
4. The main point of paper [9] is that with the previous steps, only a subset of resource links is created, that are links created between pre-scheduled activities and those activities which are immediately affected by their completion. Every activity with a *total float* (TF) different from zero therefore needs to be checked by delaying the completion time day by day and inserting a resource link when there are resource dependencies.

   Although the schedule produced in this step may be acceptable with respect to the different constraints, an additional step is executed to find alternative schedules since there may be activities that could even be delayed past their TF without affecting the project completion time.

5. Each activity that has a successor by resource links will have its resource links momentarily removed and an evaluation is performed to see if the activity can be scheduled in a period ranging from the earliest completion of all successors by resource links to the earliest LST of all the “technological” successors, without breaking resource and technological constraints. If this is the case, the previous resource links are permanently removed and new links will be created for the alternative schedule respecting the newest constraints.

The search for alternative schedules is particularly important for when the PM needs to reschedule the activities due to external factors (like equipment repair or materials delivery delay etc).
When schedule updates occur, the paper suggests the following three strategies with respect to the resource link consideration: the PM can decide to either neglect all resource links or fix them and treat them like technological relationships or to fix the resource links of only a certain duration.
The choice of a specific method is dependent on the particular project’s conditions.

Finally, the additional difference with the CPM and the RCPM is that while in the first the TF is shared for activities on the same chain, in the latter the TF may be also shared by parallel activities not belonging to the same chain. This situation is reflected by the addition of resource links between the parallel activities.

This section clarified how the concepts of float should be adapted in a resource constrained setting and how the activities can be consequently classified. These new float definitions are in turn used to evaluate an activity’s criticality and, in the end, the overall network flexibility.
Moreover the exploration of alternative schedules represents a powerful means for the project manager when he needs to perform a schedule update in an unforeseen situation, therefore it should be included and performed in the planning phase of a project’s time management.

5. Comments

The study of the methods to manage resource conflicts and their respective papers gave me the opportunity to further learn new characteristics regarding time management, including more advanced heuristics and techniques with respect to the basic version of the CPM, PERT and PDM techniques seen during the lectures.

In many practical situations resources cannot be assumed to exist in unlimited quantities and there can be many constraints that make them a shortage.
Therefore particular care must be taken when trying in scheduling the activities of a project, especially in large projects such as site construction projects.
The problem itself represents a form of an optimization problem and thus the search for the optimal solution among all the feasible solutions.
However the class of resource constrained project scheduling problems falls in the NP-hard problems and they are known to take huge computational time to be resolved.

Here the approaches are divided into two different categories: exact approaches which exhaustively check for the best available solution and heuristic approaches that employ different rules and heuristics to only visit a subset of feasible solutions and return the best one found (which can be relatively good with respect to the optimal solution).
As we have seen, many heuristic approaches seem to be an extension of some of the techniques employed in exact approaches, such as the inclusion of tie-breaking rules seen in both the Branch-and-Bound and priority rules methods.
Similarly, the basic model for heuristic approaches introduces a probability for scheduling critical activities, as also happens in sampling methods for priority rules. Here the use of a probability represents a way of attempting non-biased choices for an activity's selection.
The main difference between exact and heuristic approaches is that the exact methods are more likely to produce large instances of feasible assignments at each step, consequently rendering them computationally inefficient and also requiring large amounts of memory usage.
On the other hand, heuristic approaches not only are much faster, but also produce near-optimal solutions most of the times and they can still be applied in resource

unconstrained environments as they always generate the optimal solution in these settings.
The majority of the papers analyzed suggest that heuristic approaches are the methods to be chosen when scheduling activities for a project since the project manager almost always needs to have immediate solutions to the problem. However it may still be the case that the PM could employ exact approaches when scheduling activities of a contained project.
My personal take is that whenever there is a large project and a final schedule is needed as soon as possible, the best alternative would be to employ heuristic methods. On the other hand, if one has great computational power at disposal and the project is relatively small, the exact solution is the one I would prefer.

Nevertheless, since there exist different projects and objectives, even if someone wants to only employ heuristic methods there are several alternatives available and the best one is hard to find.

Additionally, some of the papers as [3] considered search routines for better schedules and others like [8] and [9] explicitly explored the reasoning behind the importance of alternative schedules and the event of rescheduling.
When we are in an uncertain situation for which external factors that could delay some of a project's activities are more likely to happen, methods that also provide alternative schedules are even better. Additionally, the existence of alternative schedules themself may identify the possibility of having schedules that avoid placing activities that are particularly problematic in their critical sequence, thus allowing the project manager to choose the best alternative possible and therefore strongly reflecting the importance of this practice in different scenarios.

Another comparison factor for the papers is that of the measures for the evaluation of the criticality of an activity and the flexibility of a schedule.
Paper [3] considers critical an activity whose slack exceeds a parameter *k* defined in advance while paper [5] extensively categorizes the activities as resource critical or absolutely critical and defines a criticality index for each of them based on how much it can be moved along its forward or backward total slack.

The first evaluation relies too much on the value chosen in input while the second one seems a more reasonable way to define an activity's criticality.

The other argument is related to the evaluation of the whole schedule. While the approach of paper [3] concentrates more on schedule-related costs and evaluates a schedule based on its resource costs, overhead costs and the cost of changing the resource levels, paper [7] measures the activities' scheduled free float as means of network flexibility while paper [8] uses the earliest start, latest finish and F_2 and F_3 of the alternative schedules to reflect the network flexibility.
Again, given the different objectives one has to pursue to make the project a success, the decision on how to assess schedule flexibility is highly related to the problem setting itself.

A subject of investigation is whether or not the heuristic approaches described in this review are really that "trustable". The only instance in which we find an exact quantification of the time complexity of an algorithm is when paper [4] describes serial and parallel schedule generation schemes and obtains a time complexity of $O(n^2K)$, with K the total number of resource types. This seems to be a promising result when considering heuristic approaches in general.
However, this discussion also leads to the question of checking how fitting are the resource leveling heuristics employed during the backward passes, as seen in [7], or how successful are the applied perturbations in order to find additional alternative schedules and floats as paper [8] presents.
These papers show with extensive tests that their methods indeed bring interesting results, though they certainly are not sufficient to identify the best approach.
Similarly, schedule updates as described in the latest papers still represent a point of uncertainty.

Nevertheless, the techniques and measures studied in the chosen papers should represent a powerful tool at the project manager's disposal. They can be considered a much needed support in decision making routines in project management and they can make the difference when the project manager is required to make critical decisions in scheduling the project's activities.

Moreover, a project manager that includes these methods in his employment and brings additional information such as the phantom float into consideration is more likely to earn the respect of the owner of a project given that he also incorporates resource limitations in the scheduling process to best represent the real setting.

In conclusion the analyzed papers adopt different heuristics for resolving the RCPSP and it is not very clear which of them may be the best to be employed. The truth is that there is no clear winner between them: many scheduling problems may have different settings and/or different constraints to be applied for their activities and thus some methods may be more fitting then the others.
Since the results provided by applying different methods can range in performances achieved, the project manager should not rely on the first results obtained but also try other techniques as well as softwares.

Newer approaches keep getting discovered as time passes but since it is suspected that $P \neq NP$, there will never exist a polynomial time algorithm that finds the exact solution for the RCPSP (at least for now).
Therefore we are left with the choice of employing an exact approach or an heuristic approach, although the latter is far more promising in real applications.

6. Conclusions

This essay explored the different methods employed to manage resource conflicts.
Starting from the acknowledgement of the inadequateness of established techniques adopted for time management in a project such as CPM, PERT and PDM in a practical situation, the review analyzes the possible solutions to the RCPSP.

Since previous techniques are inappropriate for scheduling resource constrained projects because they produce unrealistic schedules when resources are limited, the two possible approaches for the makespan minimization problem in a resource constrained environment are divided between exact approaches and heuristic approaches.

While exact approaches produce optimal solutions, they consume a high computational time in doing so. Therefore heuristic approaches need to be employed, especially when solving large problem instances in a short amount of time.
Nevertheless this review started by analyzing first some of the exact approaches and later moved on to the discussion of heuristic methods.

Then, several techniques have been discussed in the redefinition of floats, starting from the scheduled dates and arriving at the scheduled total and free floats.
These values have been used in identifying an activity's criticality, with the definition of an a criticality index.
Moreover, the evaluation of schedules has been performed with the above findings, identifying the flexibility of the network.

Related to the flexibility of the network was the finding of alternative, equivalent schedules each offering the same overall project duration but with different critical sequences. This can help the project manager to decide on which activities to include into the critical sequence and thus reschedule those that are particularly problematic outside of it.

The techniques analyzed in this review should therefore be part of the project manager's toolkit to make better scheduling and resource allocation decisions.
They can additionally aid in supporting the project manager in the process of reducing the potential impact of activities with a high degree of uncertainty and thus contributing to the management of a project's risk.

Appendix

In this section I summarized the notation of all the terms used in this essay.

Notation

CPM: Critical Path Method
PERT: Project Evaluation Review Technique
PDM: Precedence Diagramming Method
GERT: Graphical Evaluation and Review Technique
VERT: Venture Evaluation and Review Technique
ES: early start
EF: early finish
LS: late start
LF: late finish
FS: Finish-to-Start
SF: Start-to-Finish
SS: Start-to-Start
FF: Finish-to-Finish
AOA: activity-on-arc
AON: activity-on-node
PM: project manager
PSP: project scheduling problems
RCPSP: resource-constrained project scheduling problem
NPV: Net present value
SST: scheduled start time
SFT: scheduled finish time
LPT: latest performance time
SGS: schedule generation scheme
LFT: latest finish time
LST: latest start time
SPT: shorter processing time
MSLK: minimum slack
WCS: worst case slack
EES: earliest early start
LLF: latest late finish
TF: total float
FTS: Forward Total Slack
BTS: Backward Total Slack
F_ACA: Forward_Absolutely Critical Activity
F_RCA: Forward_Resource Critical Activity
F_NCA: Forward_Non Critical Activity
B_ACA: Backward_Absolutely Critical Activity
B_RCA: Backward_Resource Critical Activity
B_NCA: Backward_Non Critical Activity
RCACI: Resource Constrained Activity Criticality Index
RCPM: Resource Constrained Critical Path Method

Bibliography

[1] Kolisch R, Padman R, “An integrated survey of deterministic project scheduling.” *Omega 29*, 2001, pp. 249-272

[2] Kastor A, Sirakoulis K, “The effectiveness of resource levelling tools for Resource Constraint Project Scheduling Problem.” *International Journal of Project Management 27*, 2009, pp. 493-500.

[3] Wiest D. Jerome, “A heuristic model for scheduling large projects with limited resources.” 1967, pp. 359-377.

[4] Kolisch R, Hartmann S, “Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis.” *Project scheduling: Recent models, algorithms and applications*, 1999, pp. 147-178.

[5] Tormos P, Lova A, “Tools for resource-constrained project scheduling and control: forward and backward slack analysis.” *Journal of the Operational Research Society*, 2001, pp. 779-788.

[6] Patterson H. James, “A comparison of Exact Approaches for solving the multiple constrained resource, project scheduling problem.” 1984, pp. 854-867.

[7] Raz T, Marshall B, “Effect of resource constraints on float calculations in project networks.” *International Journal of Project Management*, 1996, pp. 241-248.

[8] Bowers J, “Multiple schedules and measures of resource constrained float.” *journal of the Operational Research Society*, 2000, pp. 855-862

[9] Kim K, De La Garza JM, “Phantom Float.” *Journal of Construction Engineering and Management*, 2003, pp. 507-517