CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Generalized Resource Allocation and Leveling Problem in Project Scheduling

Fayez F. Boctor
Roubila Lilia Kadri

August 2016

CIRRELT-2016-40

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2016-008.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Generalized Resource Allocation and Leveling Problem in Project Scheduling

Fayez F. Boctor*, Roubila Lilia Kadri

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, Pavillon Palasis-Prince, 2325, de la Terrasse, Québec, Canada G1V 0A6

Abstract. Many research publications dealt with the multi-mode project scheduling problem under the constraint of limited resource availability. Some others treated the problem where it is required to keep the resource utilization level as invariable as possible all over the project execution period. In real-life situations we need to simultaneously determine the amount of resources to allocate to the project during its execution and to reduce the resources utilisation variability to the minimum while trying to finish the project by an acceptable completion date. The amount of resources to allocate to the project should allow finishing the project by this date and becomes a limit on the availability of these resources at any time period. In this paper we consider this more realistic project scheduling problem where, given an upper limit on the resources that can be made available to the project and an upper limit on the project completion date, we have to determine the amount of resources to allocate to its execution, the completion date to propose to the client while minimizing the project execution cost. The execution cost is composed of two main elements: the direct cost of resources to use and the overhead cost which does not depend on the amount of allocated resources but directly proportional to the project duration. To the best of our knowledge this problem, we call the Generalized Resource Allocation and Leveling Problem (GRALP), has never been the subject of any publication before. The paper proposes a mathematical formulation of the problem as well as some heuristic solution methods. It also presents a numerical experiment to assess the performance of the proposed heuristics.

Keywords. Project scheduling, resource allocation, resource leveling, heuristics.

Acknowledgements. This research work was partially supported by grant OPG0036509 from the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

---

* Corresponding author: Fayez.Boctor@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2016

1- INTRODUCTION

Project scheduling issues have attracted a lot of attention since the pioneering work of Kelley (1963) and Wiest (1963). One of the most studied problems is the resource-constrained project scheduling problem (RCPSP) where the objective is to minimize the project duration while respecting the precedence relations between tasks and the limit on resources availability. Among the first contributions to this problem is the work by Brand et al. (1964); Wiest (1967); Pritsker et al. (1969); Fisher (1970); Davis et al. (1971) and Patterson et al. (1974).

Another important problem is called the resource leveling problem (RLP) where we may or may not have limits on resource availabilities, may or may not have a project due date and we need to schedule the execution of the project tasks while minimizing the fluctuation of the amount of resources to use. Among the first contributions in this area are the paper by Burgess et al. (1962) and the one by De Witte (1964).

In practice the project scheduling problem is somewhat different. First we do not have a fixed limit on the amount of resources. Rather we need to determine the amount of resources to allocate to the project and the allocated amount becomes the limit not to exceed. Also, we usually need to determine and to negotiate with the client a date to finish the project. So we need to determine what is the best date or range of dates to propose to the client. This date becomes then the project due date. Finally, we need to build a schedule with the least possible total cost.

This paper deals with this more realistic project scheduling problem where given an upper limit on the resources that can be made available to the project and an upper limit on the project completion date; we have to determine the amount of resources to allocate to its execution and the due date to propose to the client while minimizing the project total execution cost. The execution cost is composed of two main elements: the direct cost of the resources allocated to the project and the overhead cost which does not depend on the amount of allocated resources but directly proportional to the project duration. This problem is called hereafter the *Generalized Resource Allocation and Leveling problem* (GRALP).

In the GRALP we have a project composed of a set of tasks, a set of zero-lag, finish-to-start precedence constraint defined over the set of tasks and a set of renewable resources. There is a limit on the amount that can be made available to the execution of the project for each renewable resource. However we need to determine the exact amount that should be allocated to the project. Every activity has a number of possible execution modes where each mode is characterized by a required number of units of each renewable resource and its duration.

We assume that the direct cost of making resources available to the project is linearly proportional to both the allocated amount of resources and the project duration. In addition, there is an overhead cost which is linearly proportional to the duration of the project. This duration is to be determined and should be within a range of acceptable dates.

Thus, the objective is to determine the amount of resources to allocate to the project, to select for each task an execution mode and to determine its execution dates in order to minimize the overall project execution cost while completing it at what can be considered as an acceptable completion date.

To the best of our knowledge the GRALP has never been studied in the open literature until now. Thus we do not dispose of any method to solve it.

The remainder of the paper is organized as follows. Section 2 presents a review of some related literature. Section 3 presents a mathematical formulation of the problem and section 4 presents a numerical example to emphasize the differences between the GRALP and what is called the resource availability cost problem (RACP). Section 5 presents some heuristic approaches to solve the GRALP. Section 6 presents the numerical experience undertaken to assess the performance of these heuristics and the conclusions of this research work are given in section 7.

2- REVIEW OF RECENT AND RELATED LITERATURE

The purpose of this section is not to review the huge body of literature dealing with the resource-constrained project scheduling or with resource leveling in project scheduling but to review the most related research to the topic of this paper. Many publications review the main contributions to the RCPSP. For example, Herroelen et al (1998) provide a survey of the most important contributions made in the preceding 5 years. Kolisch et al (2001) present a survey not only of models and solution methods for the deterministic RCPSP but also survey contributions to decision support systems in this area. Herroelen et al (2005) review fundamental approaches to project scheduling under uncertainty. They review reactive scheduling, stochastic project scheduling and fuzzy project scheduling approaches. Finally, Hartmann et al (2010) give an overview of several extensions of the basic RCPSP that generalize some activity execution conditions (for example by integrating setup times or allowing for pre-emption), generalize the precedence and temporal relations (by introducing parallel execution for example) and generalize the resource constraints (by considering partially renewable resources or dedicated resources). Other, but older, surveys are given by Herroelen (1972), Davis (1973), Icmeli et al (1993), Ozdamar et al (1995), Elmaghraby (1995) and Herroelen et al (1997).

To the best of our knowledge, the GRALP has never been the subject of any publication. However, some researchers studied a problem that can be seen as related to the GRALP to some extent. Some researchers called this problem the *Resource Availability Cost Problem* (RACP) and some others called it the *Resource Investment Problem* (RIP). The objective of the RACP is to determine the amounts of resources to allocate to the project in order to minimize a resource cost function. However, we notice that this cost function is independent of the project duration. But in real life, resources cost is directly proportional to the project duration. Also we notice that, in dealing with either the RACP or RIP, the overhead cost is not taken into account. Again, in real-life situations, the overhead cost represents a significant part of the project execution cost and should not be neglected. Actually, as observed by Neumann et al (2000), the RACP is a special case of

the resource leveling problem where the objective is to minimize a weighted function of the maximum number of resource units to use.

Mohring (1984) was the first to introduce the RACP and considered the case where tasks have only one execution mode. He proposed an optimal solution procedure based on extending the partial order defined by the precedence relations in a way that respects the limit on completion time. Demeulemeester (1995) also proposed an optimal solution procedure to solve the single-mode version of the problem and called it the *Resource Availability Cost Problem* (RACP). Drexl et al (2001) developed two lower bounds on the cost function using Lagrangean relaxation and column generation techniques. The problem they considered and renamed as the *Resource Investment Problem* (RIP) is slightly different from the RACP as they did not consider any limit on the project completion date.

Heuristic approaches are also proposed to solve the RACP. Yamashita et al (2006) proposed a scatter search heuristic and Shadrokh et al (2007) proposed a genetic algorithm. All these solution methods are designed to solve the single-mode version of the problem. The only heuristic designed to solve the multi-mode version is the one proposed by Hsu et al (2005). They proposed a serial schedule scheme based on two new priority dispatching rules.

Obviously the GRALP studied in this paper is different from the RACP and the RIP. In the GRALP we minimize the total project cost including the resources usage cost and the overhead cost. Both costs are proportional to the project duration. The differences will be discussed in more details in section 4 using a numerical example.

Again, it is important to explicitly consider overhead cost as neglecting it may lead to solutions with larger project duration especially in the multi-mode case where longer modes usually require less resource amounts and consequently using such modes reduces the overall resource requirements and the corresponding availability cost. That is why we should take into account both overhead costs and resource availability costs.

3- MATEMATHICAL FORMULATION OF THE GRALP

Given the set of project tasks to execute, their possible execution modes, precedence relations, the upper bound on the amount of resources to allocate to the project, an upper limit on the project completion time, the availability cost of resources, and the project overhead cost; our problem is to determine for each task its starting date and its execution mode as well as the amount of resources to allocate to the project that minimizes the total execution cost, while satisfying the precedence relations without using more resources than what is allocated to the project, and completing the project by the required deadline.

Recall that each task has one or more execution modes and each possible mode is characterized by its duration and the amount of required resources. The total cost is composed of the resource availability cost and the overhead cost over the whole project execution time. Resource availability cost is the product of the amount of resources allocated to the project multiplied by its unit cost and by the project duration.

Before presenting the proposed mathematical formulation let us first present the notation we use to build this model.

Indices:

i task index; $i = 1, \ldots, N$

j execution mode index; $j = 1, \ldots, m_i$ (m_i is the number of possible modes for task i)

k resource type index; $k = 1, \ldots, K$ (K is the number of resource types)

t time period index; $t = 1, \ldots, H$ (H is upper limit on its completion time)

Sets:

P_i set of immediate predecessors of task i

S_i set of immediate successors of task i

A_t set of tasks that could be scheduled to be in execution at t; i.e.,

$$A_t = \{i | E_i - d_{i1} < t \leq L_i\}$$

Parameters:

f_t overhead expenses for time period t

d_{ij} duration of task i if execution mode j is used

E_i earliest finish time of task i (as given by the critical path method using the shortest execution modes)

T project earliest finish time (as given by the critical path method)

L_i latest finish time of task i (as given by the critical path method while using the shortest modes and H as a limit on completion time)

r_{ijk} number of resource units of type k required to execute task i using its execution mode j

c_{kt} availability cost of a unit of resource type k at period t

M_k maximum number of units of resource type k that can be made available to the project

Decision variables:

R_k number of units of resource type k to allocate to the project over its execution time

x_{ijt} binary variable that takes the value 1 if and only if task i is executed using mode j and finished at the end of period t

y_t binary variable that takes the value 1 if the project is in execution at period t

R_{kt} variable that takes the value R_k if the project is still in execution at period t and takes the value zero otherwise

It is obvious that y_t should take the value 1 for all periods $t \leq T$. So the model seeks to determine the values y_t of only for $t = T + 1, ..., H$. Similarly, the variable x_{ijt} should take the value zero for all periods $t < E_i + d_{ij} - d_{i1}$ or $t > L_i$. So the model seeks to determine the value of x_{ijt} for periods t such that $E_i + d_{ij} - d_{i1} \geq t \leq L_i$.

Using the above listed notation, the GRALP can be formulated as follows:

Find: $x_{ijt}; i = 1, \dots, N, j = 1, \dots, m_i, t = E_i + d_{ij} - d_{i1}, \dots, L_i$
$R_k; k = 1, \dots, K$
$R_{kt}; k = 1, \dots, K, t = T + 1, \dots, H$, and
$y_t \in \{0,1\}; t = T + 1, \dots, H$ which:

Minimize: $$\sum_{t=1}^{T} f_t + \sum_{t=T+1}^{H} f_t y_t + \sum_{k=1}^{K} \sum_{t=1}^{T} c_{kt} R_k + \sum_{k=1}^{K} \sum_{t=T+1}^{H} c_{kt} R_{kt} \quad (1)$$

Subject to:

$$\sum_{j=1}^{m_i} \sum_{t=E_i+d_{ij}-d_{i1}}^{L_i} x_{ijt} = 1; \quad i = 1, \dots, N \tag{2}$$

$$\sum_{j=1}^{m_e} \sum_{t=E_e+d_{ej}-d_{e1}}^{L_e} t x_{ejt} - \sum_{j=1}^{m_i} \sum_{t=E_i+d_{ij}-d_{i1}}^{L_i} \left(t - d_{ij}\right) x_{ijt} \leq 0; \quad i = 1, \dots, N, e \in P_i \tag{3}$$

$$\sum_{i \in A_t} \sum_{j=1}^{m_i} \sum_{s=\max\{t, E_i+d_{ij}-d_{i1}\}}^{\min\{t+d_{ij}-1, L_i\}} r_{ijk} x_{ijs} \leq R_k; \quad k = 1, \dots, K, t = 1, \dots, H \tag{4}$$

$$R_k \leq M_k; \quad k = 1, \dots, K \tag{5}$$

$$R_{kt} \geq R_k - M_k(1 - y_t); \quad k = 1, \dots, K; t = T+1, \dots, H \tag{6}$$

$$N y_t \geq \sum_{i \in A_t} \sum_{j=1}^{m_i} \sum_{s=\max\{t, E_i+d_{ij}-d_{i1}\}}^{\min\{t+d_{ij}-1, L_i\}} x_{ijs}; \quad t = T+1, \dots, H \tag{7}$$

$$y_t \leq y_{t-1}; \quad t = T+2, \dots, H \tag{8}$$

The objective function (1) expresses the total cost of the project. The first and third terms express respectively the overhead and resources availability costs of the *T* first periods; the second and forth terms give the overhead and resources costs for the periods *T*+1 up to the end of the project. Constraints (2) determine the execution mode and the end period for each task. They also make sure that each task has one and only one execution mode as well as one and only one finish period. Constraints (3) assure that any task cannot starts before the end of the execution of all its immediate predecessors and constraints (4) determine the amount of resources to be allocated to the execution of the project. Constraints (5) ensure that the allocated amounts of resources do not exceed what can be made available and the constraints (6) make sure that no resources are allocated once the project is completed. Constraints (7) ensure that y_t equals one (the project is in execution) if there are any tasks to execute at period *t* and constraints (8) implies that if the project is not in course of execution at period *t*-1, it is also not in execution at period *t*.

4- THE GRALP VERSUS THE RACP: A NUMERICAL EXAMPLE

The objective of this section is to underline the differences between the GRALP (the generalized resource allocation and leveling problem) and the RACP (the resource availability cost problem) as well as to show the effect of not considering the overhead cost explicitly in the scheduling process.

To do so, let us first present the mathematical formulation of the RACP under the assumption of multi-mode tasks. Let c_k be cost of using a unit of resource k and assume that this cost is known and constant. Then the RACP can be formulated as follows:

Minimize: $$\sum_{k=1}^{K} c_k R_k \tag{9}$$

Subject to: $$\sum_{j=1}^{m_i} \sum_{t=E_i+d_{ij}-d_{i1}}^{L_i} x_{ijt} = 1; \quad i = 1, \dots, N \tag{2}$$

$$\sum_{j=1}^{m_e} \sum_{t=E_e+d_{ej}-d_{e1}}^{L_e} t x_{ejt} - \sum_{j=1}^{m_i} \sum_{t=E_i+d_{ij}-d_{i1}}^{L_i} (t - d_{ij}) x_{ijt} \leq 0; \quad i = 1, \dots, N, e \in P_i \tag{3}$$

$$\sum_{i \in A_t} \sum_{j=1}^{m_i} \sum_{s=\max\{t, E_i+d_{ij}-d_{i1}\}}^{\min\{t+d_{ij}-1, L_i\}} r_{ijk} x_{ijs} \leq R_k; \quad k = 1, \dots, K, t = 1, \dots, H \tag{4}$$

$$\sum_{j=1}^{m_i} \sum_{t=E_i+d_{ij}-d_{i1}}^{L_i} x_{ijt} \leq H; \quad i = 1, \dots, N \tag{10}$$

Let us now consider the 5-task, 2-resource numerical example shown in Figure 1 with an overhead cost of 20 $ per time unit, resource usage cost of 2 $ and 5 $ per time unit respectively for resources 1 and 2, and a limit *H* on the project duration of 16 time units. The optimal solution of the corresponding GRALP is to allocate 3 and 6 units of resources 1 and 2 to the project which will lead to project duration of 10 time units. The resource usage cost for these 10 time units are 360 $ while the overhead cost is 200 $ for a total cost of 560 $. This optimal solution is exhibited in Figure 2.

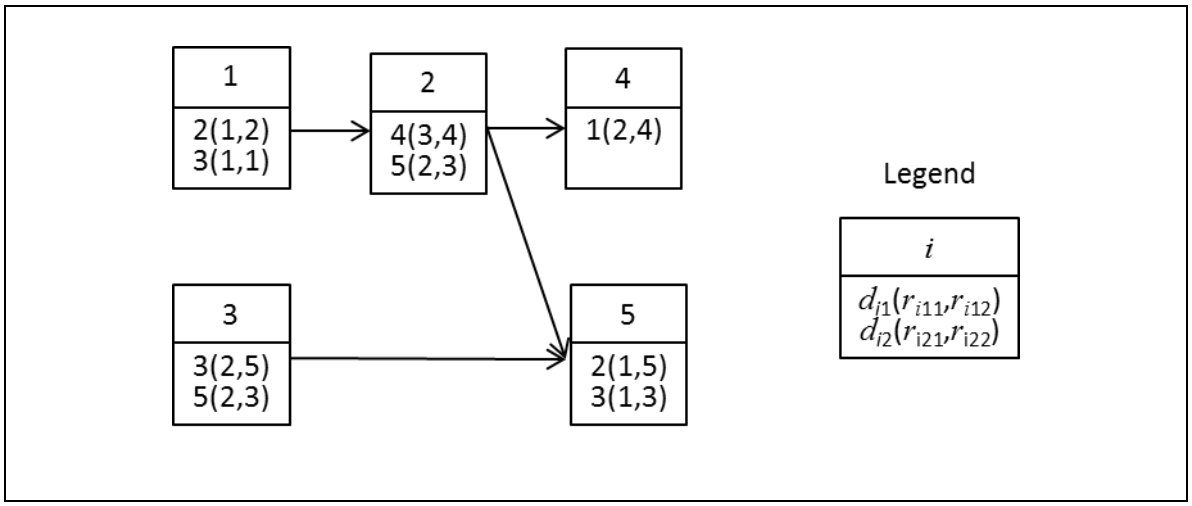

Figure 1: the 5-task, 2-resource numerical example

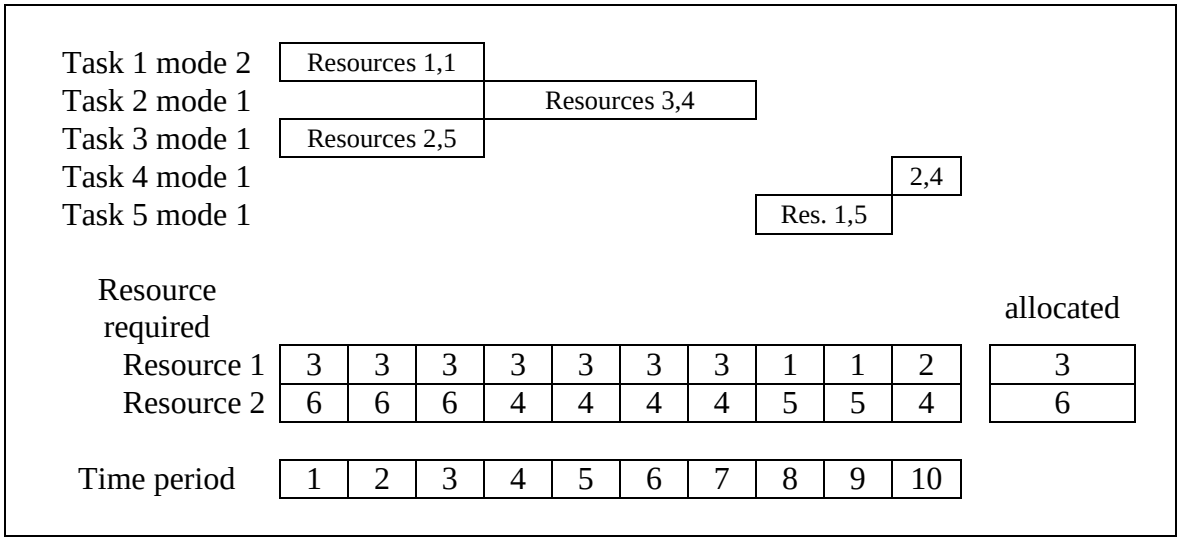

Figure 2: Optimal solution of the GRALP

On the other hand, if we handle the problem as a RACP using any multiple of 2 and 5 as availability cost (say 2Z and 5Z), the optimal duration of the project will be 16 time units and the number of resource units allocated to project will be 2 and 4 respectively. The corresponding availability cost will be 24Z. The optimal solution of this RACP is shown in Figure 3.

Clearly, as in the GRALP we consider that resource usage cost is proportional to the project duration and as we explicitly consider the overhead cost, it is not optimal to prolong the execution of the project behind a certain date. On the other hand, the optimal solution of the RACP has a duration which is either equal to the given due date or the duration that minimizes the cost of resource units allocated to the project.

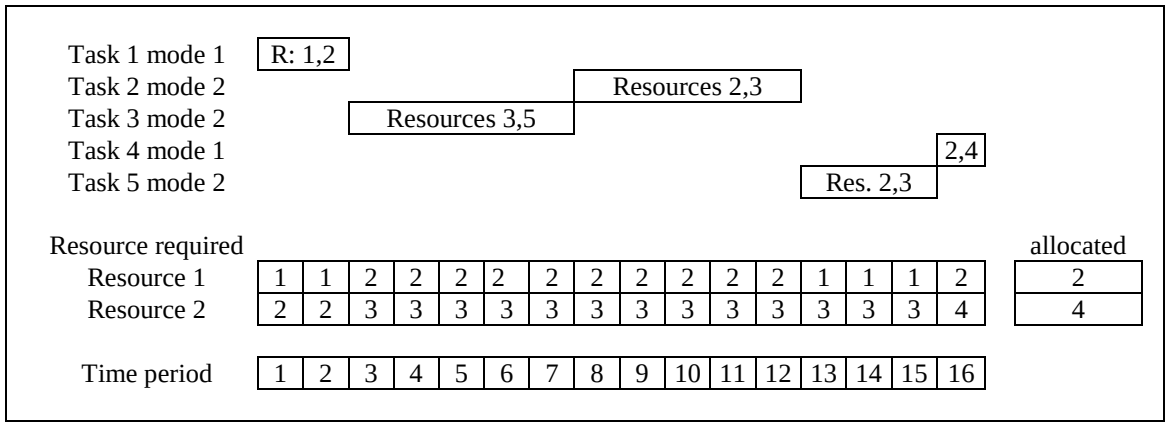

Figure 3: Optimal solution of the RACP

Now, to emphasise the importance of explicitly considering the overhead cost, let us set this cost equal to zero. In this case the optimal duration of the corresponding GRALP increases to 13 time units and we have to allocate to the project 3 and 4 units of resources 1 and 2 respectively. The resulting resource usage cost is 338 $. Obviously, as the overhead cost is neglected, the optimal duration is increased but not as much as if we assume that the availability cost does not depend on the project duration. Actually, if we extend the project duration behind 13 time units, we may require less resource units but for longer period of time resulting a higher resource usage cost.

5- THE PROPOSED SOLUTION HEURISTIC

The Generalized Resource Allocation and Leveling Problem (GRALP) is a complex one that requires making several interrelated decisions. Mainly three decisions should be made. We need: (1) to determine the number of resource units to allocate to the project, (2) to determine the execution mode for each activity, and (3) to construct an execution schedule; i.e. to determine the execution dates for the activities. Obviously these decisions are interrelated. For example the optimal execution modes and dates largely depend on the amount of resources allocated to the project and the optimal dates depend on the selected execution modes.

The heuristic approach we propose hereafter is a neighbourhood search approach where we simultaneously search three neighbourhoods each of them is related to one of these three decisions. The first decision is coded by a vector of K elements where the k^{th} element gives the number of units of resource type k to allocate to the project. The second decision is coded by a vector of N elements where the i^{th} element indicates the mode to be used to execute activity or task i. Finally, the execution schedule is coded by a vector of N elements indicating the sequence in which activities should be considered for scheduling (a priority vector). This sequence can be translated into a schedule by assigning to each activity (while using the selected execution mode) the earliest possible execution date which satisfies resource limits and precedence relations. The way to transform a sequence into schedule is straightforward and described in Boctor (1996).

The general framework of the proposed neighborhood search approach, called hereafter the 3-D neighbourhood search heuristic, is depicted in Figure 4. The details of the approach and its main procedures (called *MMRCPS*, *LNS-mode*, *RACP* and *LNS-sequence*) are explained in the remainder of this section.

Generation of allocated resources vectors

This 3-D search procedure generates a number of vectors of allocated resources $\{R_k; k = 1 .. K\}$. It starts with the vector $\{R_k = M_k; k = 1.. K\}$ and then searches its neighborhood. Actually all the vectors with $M_k \geq R_k \geq \lambda_k$; k=1.. K, except those leading to a project

duration that exceeds the duration limit *H*, will be examined. Here λ_k stands for the smallest amount of resource type *k* that should be allocate to the project. This amount λ_k is such that: $\lambda_k \geq \max_i\{\min_m r_{im}\}$. Notice that if λ_k is less than $\max_i\{\min_m r_{im}\}$, some of the activities cannot be schedule. Also $\lambda_k \geq \frac{1}{H}\sum_{i=1}^{N} \min_m\{d_{im}r_{imk}\}$ as otherwise we cannot construct a schedule with duration that does not exceed the duration target *H*. Thus λ_k can be calculated by:

$$\lambda_k = \max\left[\max_i\{\min_m r_{im}\}\ ,\ \frac{1}{H}\sum_{i=1}^{N}\min_m\{d_{im}r_{imk}\}\right]$$

Notice that in the procedure presented in Figure 4, at each iteration we reduce R_k , the allocated amount of one of the resources. However, we stop reducing the value of R_k as soon as the current value produces a schedule with a duration larger than *H*. In such a case we move to reduce the amount of the next resource type.

```
Set Rk=Mk; k=1 .. K :  R1=R1+1
Calculate λk; k=1 .. K ! The smallest amount of resource type k to allocated to the project.
Set k=0
WHILE k < K
    k = k+1
    WHILE Rk > λk
        Rk=Rk-1
        IF k > 1 THEN
            FOR r =1 TO k: Rr = Mr: NEXT r
        END IF
        Calculate earliest and latest dates as well as total slack
        Call MMRCPS ! Parallel scheduling scheme using the MINSLK rule
        Call LNS-Mode ! Local search of the mode vector neighborhood
        Call LEVEL ! A resource leveling procedure
        IF obtained duration < H THEN
            IF obtained cost < Best Cost THEN store as the best solution found
            Call LNS-sequence ! Local search on the sequence vector neighbourhood
        ELSE
            Rk = λk ! Duration larger than H is obtained. Move to reduce the amount of the next
                      resource type
        END IF
    WHILE END
WHILE END
```

Figure 4: General framework of the proposed 3-D neighborhood search approach

MMRCPS: the schedule construction heuristic

For each vector of allocated resources $\{R_k;\ k=1\ ..\ K\}$, the heuristic generates an initial mode vector and a schedule by applying a parallel priority-based scheduling heuristic that we call *MMRCPS* (multi-mode resource constrained project scheduling) heuristic. The main steps of this heuristic are given in Figure 5. The scheduling priority rule we use is the MINSLK (minimum total slack) rule. However, as the total slack of a task depends on its execution mode, the total slack is calculated for each feasible mode of the task (while setting the modes of the other tasks to the shortest one) and we use the smallest one to determine the task priority. Several researchers indicated that this rule applied within a parallel scheduling scheme produces good results (for example see Boctor 1993).

WHILE some non-scheduled tasks remain:

1- Go to the following and nearest time period where there are some available resources;
2- Establish the list of tasks that are schedulable at this time period (i.e., those for which all immediate predecessors are already scheduled and there are sufficient resources to perform them using at least one of their execution modes);
3- If the list is empty go back to step 1
4- For each schedulable task determine the list of feasible modes (i.e., modes such that available resources are larger than or equal to their resource requirements) ;
5- For each combination (task, mode) determine its total slack;
6- Choose the combination (task, mode) having the smallest total slack;
7- Reduce resource availabilities consequently and update the list of schedulable task at the considered time period.
8- Go back to step 3.

WHILE END

Figure 5: Steps of the priority-based parallel scheduling heuristic *MMRCPS*

Following the application of the MMRCPS heuristic, we obtain a mode-vector indicating the used execution mode for each task and a corresponding schedule. The schedule can then be coded by the sequence of tasks we selected in the course of this procedure.

Searching the neighborhood of a mode-vector

Without loss of generality, let us assume that, for every task, its modes are numbered in the ascending order of their duration; i.e. $d_{ij} \leq d_{i,j+1}; \forall i, \forall j < m_i$.

Now we proceed to perform a local neighborhood search to improve the mode vector. Three versions of this neighborhood search improvement procedure, called *LNS-mode*, are tested in this research work. The difference between these versions resides in their neighborhood definition. The steps of these 3 versions are given in Figures 6-a, 6-b and 6-c. The fastest one is the first one where we examine the neighborhood where only one task to which we assigned its first mode is modified and replaced by the second mode; provided that the task has more than one possible mode. In the second version we examine the neighborhood where only the mode of one task is incremented to the next mode number if it exists. In the third version the modes of two different tasks are modified where the mode of the first task is incremented to the next mode number and the mode of the second task is decremented.

```
Set improvement = TRUE
WHILE improvement = TRUE
    improvement = FALSE
    FOR i = 1 TO N
        IF mode of i =1 AND number of modes of i >1 THEN
            mode of i =2
            Call MMRCPS
            IF resulting cost < Best Cost THEN
                Save as the best solution found
                improvement = TRUE
            ELSE
                mode of i =1
            END IF
        END IF
    NEXT i
WHILE END
```

Figure 6-a: Steps of the first tested version of *LNS-mode* search procedure

```
Set improvement = TRUE
WHILE improvement = TRUE
    improvement = FALSE
    FOR i = 1 TO N
        IF mode of i < number of modes of i  THEN
            mode of i = 1+ mode of i
            Call MMRCPS
            IF resulting cost < Best Cost THEN
                Save as the best solution found
                improvement = TRUE
            ELSE
                mode of i = (mode of i) –1
            END IF
        END IF
    NEXT i
WHILE END
```

Figure 6-b: Steps of the second tested version of *LNS-mode*

```
Set improvement = TRUE
WHILE improvement = TRUE
    improvement = FALSE
    FOR i = 1 TO N
        FOR l = 1 TO N
            IF mode of i >1 AND mode of l < number of modes of l AND i ≠ l THEN
                mode of i =(mode of i) –1: mode of l =1+ mode of l
                Call MMRCPS
                IF resulting cost < Best Cost THEN
                    Save as the best solution found
                    improvement = TRUE
                ELSE
                    mode of i =1+ mode of i : mode of l =(mode of l) –1
                END IF
            END IF
        NEXT l
    NEXT i
WHILE END
```

Figure 6-c: Steps of the third tested version of *LNS-mode*

Further reduction of total cost by resource leveling

The solution obtained at the end of applying *LNS-mode* search procedure can sometimes be improved if we can further reduce the cost of the allocated resources. This can be done

by solving a resource leveling problem. This problem can be stated as follows. Given the set of tasks to perform, the selected execution mode of each task, the precedence relations between these tasks, their execution dates, and the resulting project duration, modify the execution dates in order to minimize the sum of resource requirements weighted by the resources unit costs without increasing the project duration.

Within our GRALP solution approach, this leveling problem is solved by the heuristic sketched in Figure 7 and called *LEVEL*. It consists in moving tasks having a positive free slack to start at the position that may lead to the maximum possible reduction of the resources cost, if such position exists.

```
Calculate the free slack for each task
FOR i = N TO 1
    IF free slack of i > 0 THEN
        Calculate the reduction in resource cost for each possible starting date of task i
          (search a date between current starting date and this date plus its free slack)
        Determine the position which corresponds to the maximum cost reduction
        IF maximum cost reduction > 0 THEN
            Set the starting date of i to the date leading to this maximum cost reduction
            Update the free slack of  tasks 1,2, .., i-1
        END IF
    END IF
NEXT i
```

Figure 7: *LEVEL*, the used leveling heuristic

Searching the neighborhood of the sequence vector

The next step in the proposed solution heuristic is to do a local search in the neighborhood of the task sequencing vector used to code the solution schedule. The proposed search procedure, called *LNS-sequence* search procedure, is given in Figure 8. It consists in randomly modifying the sequence vector a number of times, each time construct an initial solution using the *MMRCPS* heuristic, improve the obtained solution by applying *LNS-mode* and *LEVEL*, and calculate the cost of the resulting solution. The best obtained solution is then retained as the final solution of our GRALP.

FOR i=1 **TO** N
    Determine l_i, the position of the last predecessor of task i in the sequence
    Determine u_i, the position of the first successor of task i in the sequence
    Randomly chose a position p between l_i +1 and u_i −1 and move task i to position p
    Call *MMRCPS*
    Call *LNS-mode*
    Call *LEVEL*
    IF resulting cost < Best Cost **THEN**
        Save as the best solution found
    ELSE
        Return task i to its original position
    END IF
NEXT i

Figure 8: The *LNS-sequence* search procedure

6- PERFORMANC EVALUTATION

Ninety test instances were generated to evaluate the performance of five variants of the proposed 3-D neighborhood search heuristic. In the following we present these test instances, describe the tested variants of the proposed heuristic and summarize the obtained results.

Test instances

Three sets of 30 test instances each are randomly generated and used to evaluate the performance of the proposed heuristic. Each of these instances is composed of 30 tasks and requires 2 resource types. We limited ourselves to instances of this size as it was very time consuming and often impossible to obtain the optimal solution of larger problems. The cost of one resource unit is fixed for all instances at 2 and 5 for the two resource types respectively. All parameters values are drawn from a uniform distribution and Table 1 indicates the range of these different parameters.

Table 1: Parameters used to generate test instances

Parameter	notation	Distribution range
Number of modes	m_i	1 - 3
Number of predecessors	$\|P_i\|$	0 - 5
Task duration	d_{ij}	1 - 10
Resource requirements	r_{ijk}	0 - 9

The differences between the 3 instance sets reside in the value given to the overhead cost (f_t) and the number of resource units that can be made available to the project (M_k). For the first instance set, M_k, the number of units of resource k that can be allocated to the project, is set to equal the number of units required to schedule all tasks at their earliest possible dates while using the shortest modes. This implies that the CPM schedule is feasible with respect to the resource limits, but not necessarily optimal. Also for the first set, the overhead cost per time period is set to be the same for every time period and equals 40% of the direct cost of the allocated resources.

The overhead cost for second instance set is the same as for the first one but we use different values of M_k. To fix the values of M_k, we solved the instances of the first set to optimality using the MIP code GUROBI 5.0.1 and determined for each instance the optimal value of R_k. Let these optimal values be noted R_k^{1*}. The values of M_k for the second instance set were then set equal to $R_k^{1*} - 1$ if the $R_k^{1*}$is between 10 and 15, equal to $R_k^{1*} - 2$ if R_k^{1*} is between 16 and 20, and equal to $R_k^{1*} - 3$ if the $R_k^{1*} >$20.

The third instance set is similar to the second one except that the overhead cost is reduced by one third.

Tested heuristic variants

The results obtained by 5 variants of the proposed heuristic, referred to as H1, H2 up to H5, will be presented. However, let us mention that several other variants were tested but we report only the results of the best ones.

The first three variants H1, H2 and H3 are reduced variants of the framework presented in Figure 4 where the neighborhood search procedure called *LNS-sequence* is eliminated

and not used. This was done in order to reduce the required computation time. On the other hand, the *LNS-sequence* procedure is used within H4 and H5.

Heuristic variants H1 and H4 use the first version of the neighborhood search procedure *LNS- mode* presented in Figure 6a while variants H2 and H5 use the version exhibited in Figure 6-b. The variant H3 uses the *LNS- mode* version given in Figure 6-c.

Optimal solutions

Optimal solutions were obtained by solving the mathematical formulation given in section 3 using the MIP commercial code GUROBI 5.0.1 on a computer equipped with a 2 GHz, i7 core processor. Table 2 gives a summary of the computation times required to solve the instances of each set. This table shows that solving some instances (of this small size) may require a large computation time. This largely justifies the use of heuristic solution methods. We also notice that reducing the overhead cost leads to a significant increase in computation time.

Table 2: Computation time required to obtain the optimal solution

	Set 1	Set 2	Set 3
Minimum computation time (sec.)	202	411	691
Average computation time (sec.)	3 271	3 685	10 916
Maximum computation time (sec.)	20 565	11 443	63 626
Standard deviation (sec.)	4 695	3 054	17 242

Heuristic solutions

Tables 3, 4, 5 and 6 give the percentage increase above the optimum for the solutions obtained by the five tested variants of the proposed 3-D neighbourhood search procedure. Table 3 gives the results for the first set of instances. It shows that the average percentage increase above the optimum varies between 6.63% and 8.48%. However, the computation time required by H5, the heuristic variant that produce the smallest average deviation, is more than 40 times the computation time required by H1, the heuristic variant that produced the largest deviation.

Although the average deviation from the optimum obtained by these heuristics may seem relatively high, we have to notice that the considered problem is quite difficult to solve and involves 3 sets of decisions. We would also like to mention that we designed a genetic algorithm to solve the problem and it produced, for this first instance set, an average deviation from the optimum of 8.7% which is larger than the average percentage deviation given by any of the five tested variants of the proposed heuristic for this instance set.

Table 3: Summary of the results obtained by the proposed heuristic for instance Set 1

	H1	H2	H3	H4	H5
Average % deviation from the optimum	8.48	8.19	8.04	6.90	6.63
Minimum % deviation from the optimum	4.37	4.37	4.37	2.96	2.96
Maximum % deviation from the optimum	13.04	12.71	12.88	10.62	11.30
Standard deviation of the % deviation	2.38	2.35	2.24	1.92	2.20
Number of times the best solution is found	0	0	0	15	20
Number of times the worst solution is found	26	21	16	4	1
Average computation time (sec.)	34	46	342	1070	1364

Tables 4 and 5 seem to indicate that the deviation from the optimum solution increases when the amount of resources that can be made available to the project is reduced. The average percentage deviation from the optimum as given by the heuristic H5 increases to about 10%. However the computation time decreases significantly. This computation time reduction can be explained by the fact that the number of vectors R_k; k=1,.., K, to be enumerated by the proposed heuristic becomes smaller when the limits M_k; k=1,.., K become tighter.

Table 4: Summary of the results obtained by the proposed heuristic for instance Set 2

	H1	H2	H3	H4	H5
Average % deviation from the optimum	13.29	13.02	12.47	10.92	10.01
Minimum % deviation from the optimum	5.76	4.76	5.54	4.02	1.81
Maximum % deviation from the optimum	23.09	23.09	21.70	19.57	19.57
Standard deviation of the % deviation	4.68	4.82	4.58	4.24	4.10
Number of times the best solution is found	2	4	4	14	21
Number of times the worst solution is found	23	21	10	4	2
Average computation time (sec.)	3	4	37	91	135

Table 5: Summary of the results obtained by the proposed heuristics for instance Set 3

	H1	H2	H3	H4	H5
Average % deviation from the optimum	13.49	13.21	12.65	11.07	10.09
Minimum % deviation from the optimum	6.18	6.16	5.89	3.82	2.80
Maximum % deviation from the optimum	22.86	22.86	21.24	18.68	19.27
Standard deviation of the % deviation	4.50	4.59	4.31	4.05	3.85
Number of times the best solution is found	1	3	5	14	23
Number of times the worst solution is found	28	25	15	3	1
Average computation time (sec.)	3	4	36	94	138

Table 6 gives the aggregated results for all the 90 test instances. It confirms what we see in the previous 3 tables. In summary, all these tables show that H5 gives the best results followed by H4, H3, H2 and H1. We can also see that the average percentage increase above the optimum is inversely proportional to the computation time used by the heuristic.

Table 6: Summary of the results obtained by the proposed heuristics for all instance sets

	H1	H2	H3	H4	H5
Average % deviation from the optimum	11.69	11.42	11.00	9.60	8.85
Minimum % deviation from the optimum	4.37	4.37	4.37	2.96	1.81
Maximum % deviation from the optimum	23.09	23.09	21.70	19.57	19.57
Standard deviation of the % deviation	4.57	4.64	4.36	4.00	3.82
Number of times the best solution is found	3	7	9	43	64
Number of times the worst solution is found	77	67	41	11	4
Average computation time (sec.)	13	18	138	418	546

7- CONCLUSIONS

In this paper we introduced the *Generalized Resource Allocation and Leveling Problem* (GRALP), a new project scheduling problem that is closer to real-life project scheduling problems. In this problem we need to determine: (1) the number of units of resources to allocate to the project, (2) the execution mode to use for each activity, and (3) the execution dates for these activities. Our constraints are the precedence constraints, the maximum amount of resources that can be allocated to the project and an upper limit on the project duration. Our objective is to minimize the sum of the resource usage cost and the overhead cost.

To the best of our knowledge, the GRALP have never been the subject of any publication and consequently we do not have any previously published method to solve it. The research done in this research work seems to indicate that we are facing a difficult problem as we were not able to design a heuristic capable to produce an average percentage deviation from the optimum of better than 8.85%.

Obviously we need continue our research effort to solve this problem. Also, we need to find an approach to solve larger problems to optimality. Available MIP codes cannot allow us to solve problems with more than 30 tasks. To solve larger problem we may try to add some cuts to our model. We may also try to develop a special branch-and-bound heuristic.

We also need to develop more efficient heuristics. Developing an efficient genetic algorithm or a scatter search heuristics seems to be the alternatives to consider for solving this problem.

Acknowledgement

This research work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant OPG0036509. This support is gratefully acknowledged.

REFERENCES

Boctor FF. 1993. Heuristics for Scheduling Projects with Resource Restrictions and Several Resource-Duration Modes. *International Journal of Production Research.* 31: 2547-2558.

Boctor FF. 1996. Resource-Constrained Project Scheduling by Simulated Annealing. *International Journal of Production Research.* 34: 2335-2351.

Brand JD, Meyer WL and Shaffer LR. 1964. The resource scheduling problem in construction. Civil engineering studies report no. 5, Department of Civil Engineering, University of Illinois, Urbana, IL.

Burgess AR and Killebrew JB. 1962. Variation in activity level on a cyclical arrow diagram. *Journal of Industrial Engineering.* 13: 76-83.

Davis EW. 1973. Project scheduling under resource constraints — historical review and categorization of procedures. *AIIE Transactions*. 5: 297-313.

Davis EW and Heidron GE. 1971. An algorithm for optimal project scheduling under multiple resource constraints. *Management Science.* 17: B803-B816.

De Witte L. 1964. Manpower leveling of PERT networks. *Data Processing for Science and Engineering.* 2: 29-38.

Demeulemeester E. 1995. Minimizing resource availability cost in time-limited project networks. *Management Science*. 41: 1590-1598.

Drexl A and Kimms A. 2001. Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*. 52: 340-351.

Elmaghraby SE. 1995. Activity nets: a guided tour through some recent developments. *European Journal of Operational Research*. 82: 383-408.

Fisher ML. 1970. Optimal solution of resource constrained network scheduling problem. Technical report no. 56, Operations Research Center, Massachusetts Institute of Technology.

Hartmann, S., Kolisch, R., 2000, Experimental Evaluation of the State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research*, 127, 394-407.

Hartmann S and Briskorn D. 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*. 207: 1-14.

Herroelen W. 1972. Resource-constrained project scheduling — the state of the art. *Operational Research Quarterly*. 23: 168-173.

Herroelen W, Van Dommelen P, Demeulemeester E. 1997. Project network models with discounted cash Cows — a guided tour through recent developments. *European Journal of Operational Research*. 100: 97-121.

Herroelen W, De Reyck B., and Demeulemeester E. 1998. Resource-constrained project scheduling — a survey of recent developments. *Computers & Operations Research*. 25: 279–302.

Herroelen W and Leus R. 2005. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*. 165: 289-306.

Hsu CC and Kim DS. 2005. A new heuristic for the multi-mode resource investment problem. *Journal of the operational research society*. 56:406-413.

Icmeli O, Erenguc SS and Zappe CJ. 1993. Project scheduling problems: a survey. *International Journal of Operations & Production Management*. 13: 80-91.

Kelly J.E., Jr. 1963. The critical path method: Resource planning and scheduling", Ch. 21 in *Industrial Scheduling.* J.F. Muth and G.L. Thompson (eds.), Prentice-Hall. 347-365.

Kolisch R and Padman R. 2001. An Integrated Survey of Project Deterministic Scheduling. OMEGA *International Journal of Management Science*. 29: 249-272.

Mohring R 1984. Minimizing costs of resource requirements in project networks subject to fixed completion time. *Operations Research*. 32: 89-120.

Neumann K and Zimmermann J. 2000. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*. 127: 425-443.

Ozdamar L and Ulusoy G. 1995. A survey on the resource-constrained project scheduling problem. *IIE Transactions*; 27: 574–86.

Patterson JH and Huber WD. 1974. A horizon-varying, zero-one approach to project scheduling. *Management Science.* 20: 990-998.

Pritsker AB, Watters LJ and Wolfe PM. 1969. Multi-project scheduling with limited resources: A zero-one programming approach. *Management Science.* 16: 93-109.

Shadrokh S and Kainfar F. 2007. A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research*. 181: 86-101.

Wiest JD. 1963. The scheduling of large projects with limited resources. Ph.D. dissertation. Carnegie Institute of Technology.

Wiest JD. 1967. A heuristic model for scheduling large projects with limited resources. *Management Science.* 13: B359-B377.

Yamashita DS, Armentano VA and Laguna M. 2006. Scatter search for project scheduling with resource availability cost. *European Journal of Operational Research*. 169: 623-637.