Integration of routing into a resource-constrained project scheduling problem

Philippe Lacomme, Aziz Moukrim, Alain Quilliot, Marina Vinot

Integration of Routing into a Resource-Constrained Project Scheduling Problem

Philippe Lacomme[a], Aziz Moukrim[b], Alain Quilliot[a], Marina Vinot[a1]

[a] Laboratoire d'Informatique (LIMOS, UMR CNRS 6158), Campus des Cézeaux, 63177 Aubière Cedex, France.

[b] Sorbonne Universités, Université de Technologie de Compiègne, (Heudiasyc UMR CNRS 7253), CS 60 319, 60203 Compiègne France.

ARTICLE INFO

Article history:
Received:
Accepted:
Available:

Keywords:
Routing
Scheduling
RCPSP

ABSTRACT

The Resource-Constrained Project Scheduling Problem (RCPSP), one of the most challenging combinatorial optimization scheduling problems, has been the focus of a great deal of research, resulting in numerous publications in the last decade. Previous publications focused on the RCPSP, including several extensions with different objectives to be minimized and constraints to be checked. The present work investigates the integration of the routing, i.e., the transport of the resources between activities into the RCPSP, and provides a new resolution scheme. The two sub-problems are solved using an integrated approach that draws on both a disjunctive graph model and an explicit modeling of the routing. The resolution scheme takes advantage of an indirect representation of the solution to define both the schedule of activities and the routing of vehicles. The routing solution is modeled by a set of trips that define the loaded transport operations of vehicles that are induced by the flow in the graph. The numerical experiments prove that the models and the methods introduced in this paper are promising for solving the RCPSP with routing.

1 Introduction

1.1. Resource-Constrained Project Scheduling Problem

The Resource-Constrained Project Scheduling Problem (RCPSP) represents a challenging research problem that has been widely studied over the past decades. This problem is composed a set of activities, $V = \{0, \ldots, n+1\}$, with durations, $p = (p_0, \ldots, p_{n+1})$, where n is the number of non-dummy activities. All this activities define the project. There are two dummy activities, 0 and $n+1$, such that $p_0 = p_{n+1} = 0$. These activities 0 and $n+1$ correspond to the "project start", which is the predecessor of all activities, and to the "project end", which is the successor of all activities, respectively. The set of non-dummy activities is identified by $A = \{1, \ldots, n\}$. The activities are linked by two kinds of constraints, the precedence constraints (one activity j cannot start before all its predecessors have been achieved) and the resource constraints induced by the resource exchanges (an activity requires resources to be achieved). A schedule of the RCPSP can be represented as a vector of activity start times, $S = (S_0, \ldots, S_{n+1})$, where $S_i \in \mathbb{N}$, with the associated vector of activity completion times, $C = (C_0, \ldots, C_{n+1})$. The precedence graph is denoted $G = (V, E)$, where nodes in V are activities and edges in E are precedence relations. For each activity $i \in V$, all outgoing arcs $(i, j) \in E$ are weighted by its duration p_i. If there are arcs $(i, j) \in E$, then $C_i = S_i + p_i \leq S_j$ since activity j has to be scheduled after activity i. Each activity requires some amount of renewable resources. The number of project resources is denoted as q and the set of resource is $R = \{R_1, \ldots, R_q\}$. The activity resource requirement $b_{ik} \in \mathbb{N}$ means that activity i requires $b_{ik} \leq B_k$ resource units of resource k during its execution such that B_k denotes the availability of resource k. The longest path from 0 to $n+1$ in graph $G(V, E)$ corresponds to the critical path.

Because the RCPSP is an extension of the job-shop, it is an NP-hard problem (see Blazewicz et al. (1983) and Brucker et al. (1999) for details on complexity). Several surveys are available including, but not limited to, the survey of Herroelen et al. (1998), of Brucker et al. (2000), of Kolisch and Padman (2001), of Weglarz (1999) in the book on Project Scheduling, of Demeulemeester and Herroelen (2002) and of Hartmann and Briskorn (2010). Note that Brucker (1999) provided a classification scheme for the relationships between RCPSP and machine scheduling. More recently, Kolisch and Padman (2001) introduced a survey of heuristic methods for the different classes of project scheduling problems.

[1] Corresponding author. e-mail addresses: placomme@isima.fr (P. Lacomme), aziz.moukrim@utc.fr (A. Moukrim), alain.quilliot@isima.fr (A. Quilliot), marina.vinot@isima.fr (M. Vinot)

Numerous models and several formulations have been introduced including a formulation based on forbidden sets introduced by Alvarez-Valdés and Tamarit (1993), the aggregated time indexed formulation (Pritsker et al. 1969), the time-indexed formulations with step variables of Pritsker and Watters (1968), the compact event-based formulation that has been used to address scheduling problems with machines (for example, Dauzère-Pérès and Lasserre (1995)), and the recent flow formulation of Artigues et al. (2003).

A RCPSP solution can be linked to the network flow theory (Ahuja et al., 1993) since it involves circulation of resources that can be goods, people or energy, for example. RCPSP resolution based on flow approaches received attention in Artigues and Roubellat (2000) and Artigues et al. (2003). As stressed by Artigues et al. (2003), a solution of the RCPSP can be modeled by an activity-on-node (AON)-flow network where the earliest starting time (S) of activities is computed using a longest path algorithm and where disjunctions are resolved by a flow definition. An AON-flow network representation permits to model the circulation of resources in a graph where nodes correspond to activities and arcs correspond to resources transferred (Artigues et al., 2003).

1.2. Problem of interest: RCPSPR

The RCPSP with routing (RCPSPR) is an extension of the RCPSP where the resource transport between activities must now be achieved by a limited fleet of vehicles. The problem consists of finding a feasible schedule of minimal duration C_{max} by assigning a starting time to each activity (classical RCPSP) as well as assigning a vehicle to each transport of resources with a starting time such that the precedence relations, the resource availabilities and the vehicle capacities are respected. Such problems occur in practice when a resource is physically moved from one location to another (e.g., when a crane has to be transported between construction sites or for scaffolding moved between sites to complete a project). In contrast to the classical RCPSP, no article appears to have dealt with this extension so far. This paper focuses on the case of one resource $q = |R| = 1$ and, therefore, $b_{ik} = b_i$.

The routing problem consists of scheduling trips for a set of vehicles $T = \{1, \dots, v\}$ sorted in descending order of capacity $c_u, u \in \{1, \dots, v\}$ with a loaded transportation time t_{ij}^{ux} from activity i to j with a vehicle u loaded with x units of resources and an unloaded transportation time e_{ij}^{u} from activity i to j. Since the transportation times are vehicle-independent and vehicle load-independent, $t_{ij}^{ux} = t_{ij}$ and $e_{ij}^{u} = e_{ij}$. The vehicle is responsible for the transport (transfer) of the resource. An activity j is defined by a starting time S_j with a completion time $C_j = S_j + p_j$ and resource supplies that meet the requirement b_j. An activity can only start when a total amount b_j of resources has been transferred from activities to activity j. Each transport operation $T_{(i,j,u,x)} = \left(P_{(i,j,u,x)}, D_{(i,j,u,x)}\right)$ from activity i to j is fully defined by a pickup operation $P_{(i,j,u,x)}$ and a delivery operation $D_{(i,j,u,x)}$. A pickup operation and a delivery operation are defined by:

- A departure time and an arrival time of the vehicle, $B_{(i,j,u,x)}$ and $A_{(i,j,u,x)}$, respectively;
- A quantity of resource (pickup or delivery) x;
- A vehicle u assigned to the transport.

In this paper, a transport operation, $T_{(i,j,u,x)}$ is a couple of pickup/delivery operations and $B_{(i,j,u,x)}$ is the departure time of the vehicle (starting time of the transport operation) (Fig. 1). A similar remark holds for the arrival time $A_{(i,j,u,x)}$ of the vehicle u on activity j. The earliest starting time of one activity S_j depends on the arrival time of all vehicles, i.e., of all delivery operations.

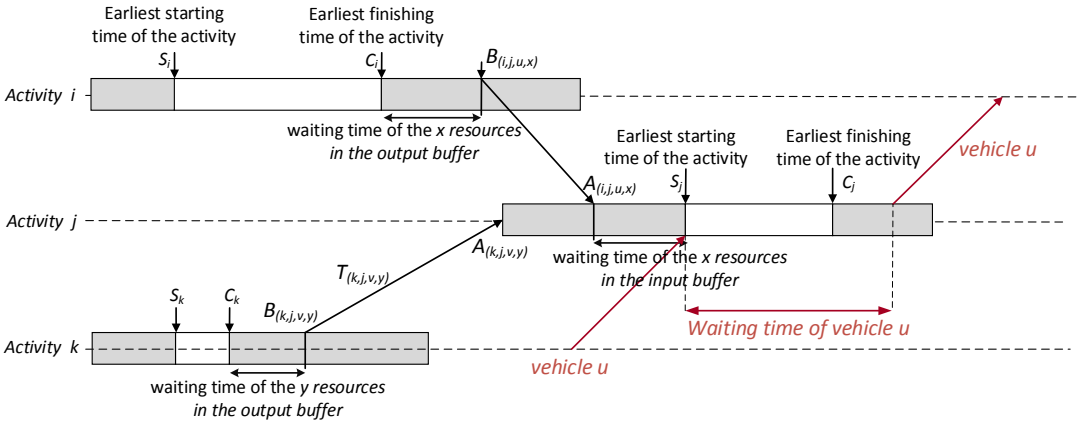


Fig. 1. Coordination between transport operations and activities

The difference between S_j and $A_{(i,j,u,x)}$ defines the total waiting time of the x units of resource in a buffer (Fig. 1). The same remark applies for the resources after the completion of one activity. The time windows defined

between the completion time C_i and the departure time $B_{(i,j,u,x)}$ models a waiting time of resources in an output buffer.

The total amount of resources available upstream of activity i is an increasing function (which depends on the delivery operations) before the starting time of activity i, and a decreasing function that models the remaining resources after completion of i and before the pickup operations (Fig. 2).

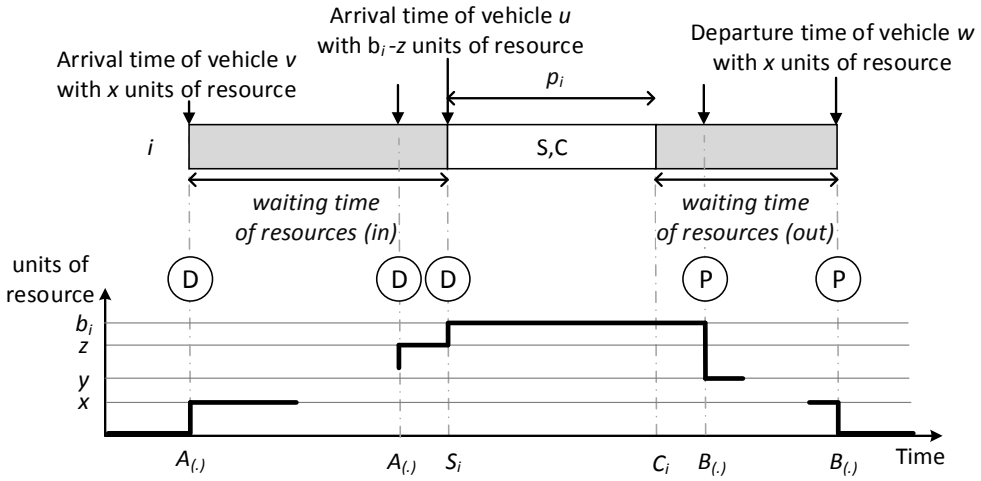


Fig. 2. Resource transfers for an activity

The vehicles are also submitted to waiting times between two successive transport operations. Because no buffer constraints are addressed, the waiting time of a vehicle depends only on the activity finishing times and is defined by the difference between the arrival time of the vehicle on one activity (to deliver some resources) and the vehicle departure time (which is equal to the activity finishing time), as can be seen in Fig. 2. The vehicle can immediately leave after a delivery operation on an activity, the resources are stored or used by the activity. To reduce the waiting times, an integrated approach should be developed to solve the problem, with the scheduling, the routing and the assignment of vehicles for each transport operation. Nevertheless, the waiting times are not taken into account explicitly in the objective function in this paper, and the objective function addresses the makespan minimization. In this paper, the resources are not allowed to make trips and only direct resource transfers are permitted to address routing problems with perishability or security (service quality) constraints where mistakes concerning vehicle unloading operations must be fully eliminated.

1.3. Classification

Numerous classifications of resource types have been introduced for years by Brucker et al. (1999) and Demeulemeester and Herroelen (2002) and make a strong distinction between renewable, non-renewable, and partially renewable resources. Taking advantages of the classification recently defined by Krüger and Scholl (2009), the problem concerns a problem with:

- A 1st tier resource that must be transferred at a certain time and that models a personnel, a tool, a material or any heavy machinery required to achieve an activity.
- A 2nd tier resource that supports the transfer of the 1st tier resource.

The type of 2nd tier resource transfer is assumed to be a physical transfer that is characterized by moving the 1st tier resource from one location to another. The physical transfer is assumed to be achieved by a material handling resource denoted below the vehicle (without loss of generality) so as to be consistent with the widespread current usage in routing and a transfer defined as a transportation problem. Consequently, a RCPSP satisfying the previous definition can also be denoted RCPSPR (with Routing) since only physical transfers are addressed for 1st tier resources by 2nd tier resources.

2 Scheduling/Routing context

In recent years, a tremendous amount of research has been devoted to production and transportation sub-problems and, as highlighted by Chen (2010), integrated production and transport models at a detailed scheduling level are fairly recent and the majority of models attempts to jointly optimize operation-by-operation production and transport. The RCPSP with routing (RCPSPR) falls into the category of production and transportation scheduling

problems (PTSP), which are key operational functions in a supply chain since it is critical to jointly integrate these two planning and scheduling functions in a coordinated manner.

2.1. Modeling transport in the scheduling problem

Several scheduling problems encompass some extensions with transportation constraints involved by critical handling resources, including, for example, the flexible job-shop scheduling problem with transport (see Zhang et al (2012)), the job-shop with transport (see Knust (1999), Lacomme et al. (2007), Afsar et al. (2016)), the Flexible Manufacturing Systems (FMS) (Caumond et al., 2009), the HSP (Hoist Scheduling Problem) (Honglin et al., 2016; Adnen and Mohsen, 2016; Chtourou et al., 2013) or the RCPSP with transport (see Quilliot and Toussaint (2012)). Two recent papers (Weiss and Schwindt, 2016; Poppenborg and Knust, 2016) addressed an integrated scheduling and routing problem with time transfer constraints in RCPSP-like problems. However, in these studies, the routing is not explicitly modeled with a fleet of vehicles.It is commonly accepted that most of the scheduling approaches take advantage of a modeling based on a graph that is derived from the disjunctive graph of Roy and Sussmann (1964). This model has been extended to scheduling problems with transport (for the job-shop by Lacomme et al. (2007) and Zhang et al. (2012)), where transport operations are modeled by a vertex with disjunctive arcs between each operation that requires the same resource (vehicle).

The integration of transport constraints into scheduling problems can be modeled in two possible ways considering two situations. The first one consists of physically moving resources from one location to another (modeling heavy-machine transport or specialists flying around the world, for example). The second one consists of models based on time-lags between machines where time-lags are a variant of transfer time. Time-lags are associated with activities, whereas setup times are associated with resources. It should be noted that setup times can depend on both the schedule and on the resource management.

Minimal time-lags between activities are commonly used to model a minimal delay and can depend on the consecutive activities. If explicit vehicle management is not required, this approach provides an efficient modeling of transport and can also be relevant in problems where the vehicle fleet is large enough to assume that a transport can be achieved with no delay after the end of one activity. The problem studied by Poppenborg and Knust (2016), referred to as RCPSP with transfer times belongs to this approach. The integrated problem studied in this paper extends this recent paper by taking vehicle management into account.

Maximal time-lags between activities can lead to a maximal delay for the transport and are commonly used in pickup and delivery problems where a maximal riding time is defined. It is important to note that two widely used approaches for trip evaluation introduced first in Cordeau and Laporte (2003) and, second, in Firat and Woeginger (2011) take advantage of such modeling approaches, as reported in Chassaing et al. (2015).

Explicit transport modeling depends on the vehicle capacity and, in the specific situation of unary vehicle capacity (which means that the vehicle cannot achieve more than one transport operation at a time), a trip is a sequence of operations defined by a Pickup operation (P), a Delivery operation (D), a Vehicle (V) and a quantity transported (q), i.e., by a quadruplet (P, D, V, q). Hence, a solution of the routing problem consists in defining the disjunctions between the transport operations (square node in Fig. 3).

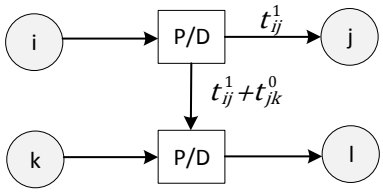


Fig. 3. Explicit representations of transport operations with a single capacity vehicle

Depending on the problem, the transportation time can be job-dependent and vehicle load-dependent and can be denoted t_{ij}^x. A similar notation holds for the unloaded vehicle transport denoted t_{ij}^0 (commonly denoted $e_{ij} = t_{ij}^0$).

As stressed by Lacomme et al. (2013), the disjunctive arc between two transport operations is modeled by an arc with value $t_{ij}^1 + t_{jk}^0$, which corresponds to a loaded transport from i to j and an unloaded transport from j to k Fig. 3.

Such approaches have been used in the flexible flow-shop resolution by Zhang et al. (2012) and in the job-shop by Lacomme et al. (2010). For the RCPSP-like problem, as pointed out by Krüger and Scholl (2009), resource transfers between activities are known to be highly relevant in practice but most research papers neglect them.

2.2. Routing in scheduling problems

Previous related studies only focused on transport with scheduling (including the job-shop, for example) since the transport was only used to check some extra constraints in scheduling. However, no previous study exists where the transport is modeled using classical approaches taken from the routing community. It is unfortunate that

classical routing approaches are not widely used in integrated problems with scheduling, in view of the huge number of efficient routing approaches that tackle routing with some extra constraints. Numerous approaches to routing problems take advantage of indirect solution representations and are consistent with the same trend of widespread approaches in scheduling. Many publications use a routing solution model based on giant tours as well as a split-based approach to transform a giant tour into one routing solution. The split approach was first introduced by Beasley (1983) as the second phase of a route-first, cluster-second heuristic to solve the Capacitated Vehicle Routing Problem (CVRP). The first phase computes a giant tour of all customers (solving a Traveling Salesman Problem (TSP)) by relaxing vehicle capacity and maximum tour length. The second phase constructs a cost network in which an arc represents a feasible trip of the CVRP visiting a subsequence of the TSP tour, and then applies a shortest path algorithm to find least cost feasible trips. The principle was then further used as of 2001 when it was implemented within a more general framework for routing problems for the Capacitated Arc Routing Problem (CARP) (Lacomme et al., 2001; Lacomme et al., 2004), and the method has been the best published method from 2001 to 2008 for the CARP. After this, indirect approaches for routing were more commonly used. Prins (2004) introduced a very efficient method for the VRP that also took advantage of the split algorithm. In this context, the number of split applications in routing problems strongly increased, as pointed out by Duhamel et al. (2011) and Prins et al. (2014), and now covers CARP, VRP, location routing and numerous extensions that represent a set of more than 70 publications.

3 Definition of a RCPSPR solution

3.1. Example

The Gantt chart in Fig. 4 defines a schedule by defining the starting times of activities and by defining the trips of vehicles that are composed of an ordered set of pickup and delivery operations. Activity 4 is scheduled from time 10 to 40 (Fig. 4), and the earliest finishing time (with a value of 40) of activity 4 defines the time when the resources used by activity 4 are free to be transported from their current position (in the unloaded buffer upstream of activity 4) to a new activity. Vehicle 1 is assigned to transport resources from the depot to activity 4 with two units of resources and from activity 3 to 1 with two units of resources. Between these two successive loaded transports with two pickup/delivery operations, the vehicle achieves an unloaded transport from the position of activity 4 to the position of activity 3.

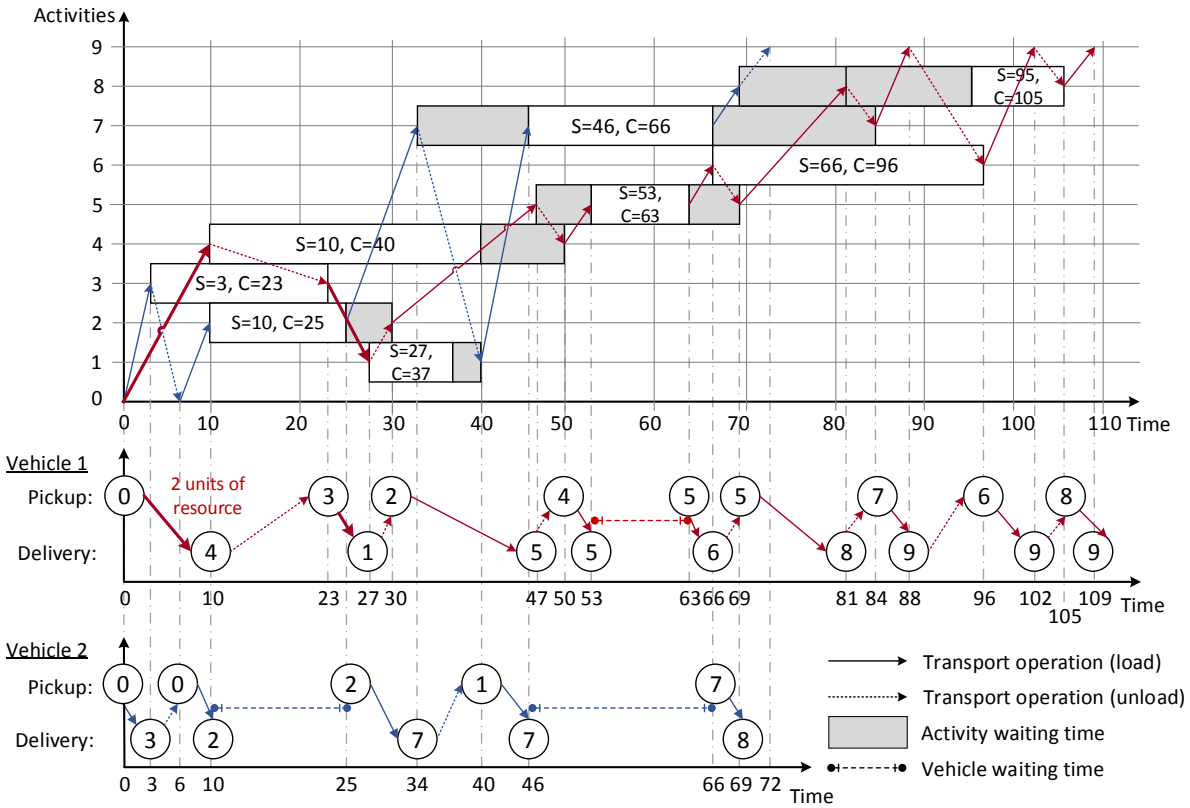


Fig. 4. Gantt solution with both scheduling and transport operations

Consequently, a full description of a vehicle trip is an ordered sequence of loaded transport operations (pickup/delivery operations) and of unloaded transport operations. The first transport operation is a loaded transport operation from depot 0 to activity 4 with a vehicle departure time $B_{(i,j,u,x)} = 0$ and a vehicle arrival time $A_{(i,j,u,x)} = 10$ with a load of two units of resources. Between operations 7 and 8 in trip 1, a waiting time of the vehicle in activity 5 can be observed.

A full description of this example is available at: http://fc.isima.fr/~vinot/Research/RCPSP&Routing.html.

3.2. Modeling a solution

In the previous section, a routing solution was defined as a set of trips and each trip was defined by an ordered sequence of transport operations. Instead of considering both loaded and unloaded transport operations, it is possible to consider only the loaded transport operations to fully define a trip. Indeed, an unloaded transport occurs between two loaded transports, and because there is no cost for the waiting time of resources in buffers, the unloaded transport operation can be scheduled at the earliest starting time. For the sake of convenience, a loaded transport operation is referred to as "transport operation" in the following sections and is defined using a quadruplet (pickup operation, delivery operation, vehicle assigned, quantity of resources transferred).

With the hypothesis of the direct transport of resources between activities with respect to the quantity of resources required (no more, no less), it is possible to use a flow model (Artigues et al., 2003) to determine the transport operation defined by one pickup operation achieved at activity i and by a delivery operation achieved at activity j with a flow φ_{ij}. A transport operation (i, j, u, φ_{ij}) is fully defined by the departure time $B_{(i,j,u,\varphi_{ij})}$ of the vehicle u and by the arrival time $A_{(i,j,u,\varphi_{ij})}$ of the vehicle u with φ_{ij} units of resources. For the sake of convenience, $(i, j, *, \varphi_{ij})$ refers to a transport operation for which the assignment of a vehicle is not yet defined. Transportation creates new constraints on the starting time of the activity since S_j, the starting time of activity j, must be greater than all vehicle arrival times that transport resources to activity j.

Since the objective function, commonly defined in job-shop, flow-shop or in FMS, consists in makespan minimization (i.e. in the minimization of the earliest finishing time of all operations) the RCPSPR is addressed in the same trend. Because there is no distinction between the riding time cost of one vehicle and the total routing time cost, only the total transport duration is addressed and only semi-active solutions are required, which means that only left-shifted routing solutions can be investigated. A semi-active routing solution computation implies that no waiting time of vehicles is possible on the pickup node (node where a pickup operation is achieved) of one transport since all transports are scheduled at the earliest possible time. Therefore, the waiting time of vehicles is assumed to be located at the delivery node (node where a delivery operation is achieved) only.

To conclude, a fully oriented disjunctive graph that models a solution can be obtained by assigning vehicles to transport operations and by a definition of the disjunctions between operations. This graph does not encompass any activity disjunction except disjunctions of activities defined in the problem.

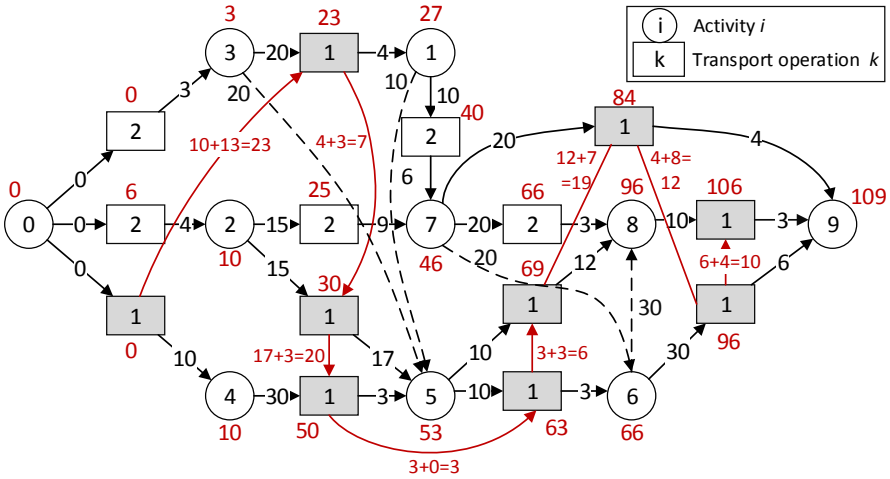


Fig. 5. Example of a solution with a representation of the transport disjunctions limited to vehicle 1

The combinatorial optimization problem is composed of the assignment of vehicles to transport, the definition of disjunctions between all transport operations assigned to the same vehicle, and of the earliest starting times for

both transport operations and activities (nodes in the graph), which can be obtained by any longest path algorithm. Figure 5 provides an oriented disjunctive graph composed of two trips assigned to the vehicles.

To obtain a readable graph, only disjunctions of vehicle 1 are included in the graph, defining a sequence of transport operations:

- First: from activity 0 to 4 with a departure time with a value of 0;
- Second: from activity 3 to 1 with a departure time with a value of 23. This transport operation is in disjunction with the first one, with a disjunctive arc with a cost of 23 units of time;
- Third: from 2 to 5 with a departure time of 30;
- And so on.

The sequence of disjunctive arcs defines the trip of the vehicle and ensures the coordination between activities and transport operations by left-shifting both to the earliest starting times. For example, activity 4 immediately starts at time 10 ($S_4 = 10$) since the earliest vehicle arrival time at node 4 is exactly equal to 10. The earliest departure time of the vehicle at node 3 (pickup operation of the second transport operation) is equal to $10 + 13 = 23$ where 13 is the unloaded transport operation duration from activity 4 (delivery of transport operation 1) to 3 (pickup operation of transport operation 2).

The approach advocated in Section 4 takes advantages of the disjunctive graph, making it possible to take the following factors into consideration:

- Definition of disjunctions between activities due to precedence constraints;
- Definition of at least one transport to each flow;
- One assignment of a vehicle to a transport operation;
- Definition of disjunctions between transport operations;
- Evaluation to obtain the earliest starting times of transport operations and activities.

This modeling is integrated into a hybrid metaheuristic based on a GRASP×ELS with an indirect representation of the solution.

4 A resolution scheme based on an AON-flow network extension

The key point for the resolution of the RCPSPR is to alternate between solutions encoded as a sequence of activities and a solution of both routing and scheduling. This strategy represents a straightforward continuation of the policies used in numerous routing problems, including but not limited to the CARP, the VRP and the LRP, as reported in the recent survey of Prins et al. (2014).

4.1. Proposition for a GRASP×ELS framework

The proposition is based on a GRASP×ELS (Prins, 2009), which is a hybridization of a GRASP (Greedy Randomized Adaptive Search Procedure) (Feo and Resende, 1995) with an ELS (Evolutionary Local Search) (Wolf and Merz, 2007), taking advantage of both methods. The framework uses three key points:

- An algorithm to generate an activity sequence w presented in Section 4.2;
- A procedure to evaluate an activity sequence w considering a parameter δ in order to obtain a solution of the RCPSPR, in Sections 4.3 to 4.8;
- A local search to improve an RCPSPR solution, in Section 4.9.

The parameter δ makes it possible to tune the resource capacity $R_1/R_1 -$ r with $\mathrm{r} \in [0, \delta]$ and $\delta = R_1 - \max_i b_i$, to consider different levels of resources consistent with the resource capacity. The number of GRASP iterations takes this parameter into account and is equal to $np \times (\delta + 1)$.

4.2. Generation of a sequence: indirect representation of solutions

Activity sequence w can be defined as a topological order of activities that makes it possible to compute a solution for both routing and scheduling problems, consequently defining a one-to-one indirect representation of solutions (Chen et al., 1996) (Fig. 6). An activity sequence w can be obtained considering a classification of activities based on the longest path that concerns the number of arcs.

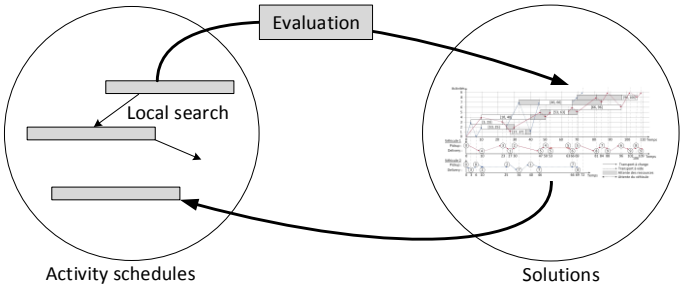


Fig. 6. Indirect representation of solutions

Activities with the same maximal distance from the dummy start activity 0 are gathered at levels where level l_m encompasses all activities with a maximal distance m from the dummy start activity. Therefore, activity 0 is at level l_0 and activity $n+1$ is at level l_{max}, where subscript max corresponds to the last level number. A final feasible sequence can be created from levels such that $W = (l_0 \dots l_{max})$ and additional feasible sequences can be created by shuffling the activities on the same level.

No violation of precedence constraints can be reported considering $w \in W$ and it is possible to define permutations in w that lead to a feasible sequence. Move $swap(u, v)$ defines a feasible move if there is no edge modeling a precedence constraint at indices $u+1$ to v.

4.3. Evaluation of a solution from $w \in W$

Roughly speaking, the evaluation procedure assign a solution (of both scheduling and routing) to one $w \in W$ in five steps:

- **Solving the flow problem** to obtain an acyclic graph encompassing both arcs for precedence constraints and arcs for the flow, in order to define the transport operations with the quantity of resource;
- **Defining a set of batch transport operations** according to the flow and the vehicle capacity to reduce the search spaces in the next steps (a batch transport operation is a set of transport operations between the same activities. A more formal definition is introduced in section 4.5);
- **Creating one giant tour** that models an ordered sequence of transport operations defined in the previous step, from one pickup operation to a delivery operation;
- **Splitting** the giant tour into trips that provide the routing and the scheduling simultaneously.
- **Evaluating** the disjunctive graph to obtain a RCPSPR solution.

Contrary to classical approaches in scheduling or in routing (where a giant tour is transformed into trips thanks to only one function), the assignment of one solution to one activity sequence w is a five-step procedure (Fig. 7) requiring successive resolutions of one flow problem, pre-calculation of the batch transport operation, generation of a giant tour and the splitting of the giant tour into trips with an evaluation to simultaneously address scheduling and routing constraints. After the evaluation procedure, a local search procedure can be applied to transform a solution of RCPSPR into a local minima.

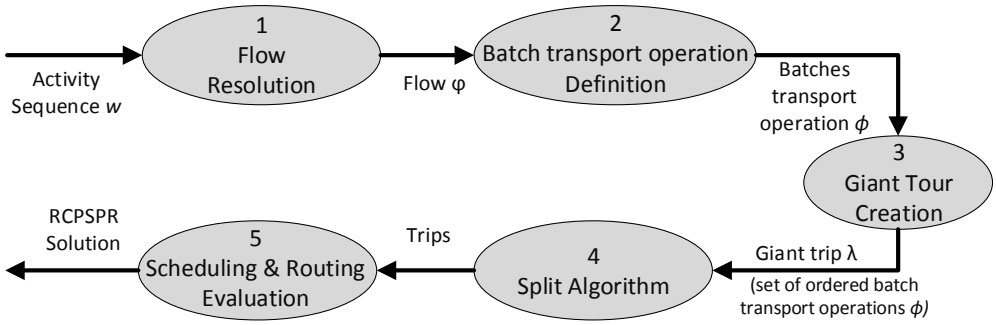


Fig. 7. The evaluation procedure applied to obtain a RCPSPR solution

4.4. Flow definition (first step of the evaluation)

Because the problem cannot be efficiently divided into sub-problems considering the flow problem first and the routing second, there is no possibility of providing a quality flow solution since the criteria to optimize cannot be taken into account during the flow resolution. Nevertheless, it is possible to use time efficient approaches to provide flow solutions. In the next sections, numerical experiments prove that this assertion is relevant since the optimal flow solution (flow providing the minimal scheduling finishing time for the RCPSP) does not lead to the optimal RCPSPR one.

Because the solutions are encoded as a sequence of activities, the scheduling decision consists in defining the flow $\varphi_{w(i)w(j)}$ (flow from node $w(i)$ to node $w(j)$; Fig 8) satisfying the activity resource requirement with the flow transfer. The flow conservation at every node is required, except for dummy nodes 0 and *n+1*. The output graph $G_\varphi(w)$ is the graph, with a flow φ satisfying the precedence constraints induced by the feasible sequence w. To obtain an efficient computation time algorithm (avoiding a costly repair procedure), it is only possible to create an acyclic graph by the introduction of a constraint in the flow linked to the order of the sequence of activities ($\forall j < i,\ \varphi_{w(i)w(j)} = 0$). Taking advantage of the special structure of the graph, it is possible to provide heuristic resolution schemes offering a flow on a preferential basis to the closest activities.

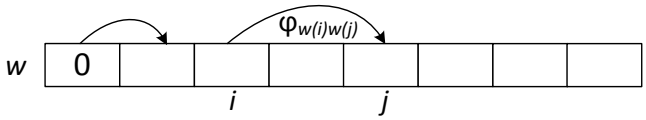


Fig. 8. Constraints on flow

The first loop from step 11 to 21 (in Algorithm 1) iterates on the activities considering the remaining amount of resources available after the process of activity $w(i)$ i.e. $b_{w(i)}$. The second loop scans all successive activities considering the maximal quantity of resources that can be transferred from $w(i)$ to $w(j)$. At each iteration, the remaining resources quantities r and l respectively for activities $w(i)$ and $w(j)$ are updated considering $v = min\,(r, l)$.

```
   Procedure name: Flow Definition
1.  procedure Flow_Definition
2.  input parameters
3.    w: activity sequence
5.  output parameters
6.    φ: a flow
7.  global parameter
8.    n: number of activities
9.  begin
10.    i:=0
11.    while (i < n)
12.     if (i = 0) r = B else r = b_w(i)
13.     stop = 0; j = i+1;
14.     l := b_j
15.     while ((stop=0)and (j ≤ n))
16.       v := min(r, l)
17.       r := r-v ;  l := l-v
18.       if (r = 0) stop = 1 else j := j + 1
19.     endwhile
20.     i := i + 1
21.   endwhile
22. end
```

Algorithm 1. Flow Definition procedure

This greedy approach, applied to the sequence of activities $w = [0, 3, 2, 4, 1, 5, 7, 6, 8, 9]$ for a total amount of six units of resources available on node 0 ($B_0 = 6$) provides the solution in Fig. 9.

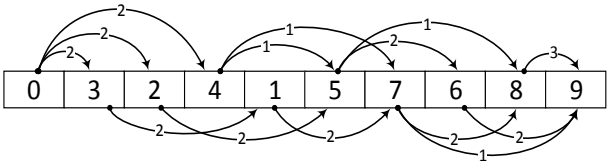


Fig. 9. Example of flow solution

4.5. Batch transport operation definition (second step of the evaluation)

With the flow created in the previous steps, all the quantities that must be transferred between activities are defined. The flow φ_{ij} created between activity i and activity j defines a batch flow transport operation with only one transport operation $(i, j, *, \varphi_{ij})$ where φ_{ij} could exceed the vehicle capacity and may require several transport operations due to the vehicle capacity constraints (c_v). For one batch flow transport operation, it is possible to define different batch transport operations $\phi_{ij} = \cup_x (i, j, *, k)_x \ / \ \sum_x k_x = \varphi_{ij}$ with different properties. In other

words, a batch transport operation ϕ_{ij} is an ordered set of transport operations from activity i to j, and each transport operation $(i,j,*,k)$ is fully defined by the quantity of resource to transport.

For example, a unitary batch transport operation $\phi_{ij} = \cup_x(i,j,*,1)_x$ defines a set of transport operations that can be assigned to any vehicle. For a non-unitary batch transport operation $\phi_{ij} = \cup_x(i,j,*,k)_x$, a transport operation $(i,j,*,k)$ can be assigned only to one vehicle u of capacity $c_u \geq k$.

The definition of a batch transport operation (set of transport operations $(i,j,*,k)$) makes it possible to define a giant tour and to then create the trips with the split procedure. The quantity of resources in each transport operation $(i,j,*,k) \in \phi_{ij}$ has an impact on the split execution since the split procedure is devoted to the aggregation of transport operations (see Prins (2004)). Unprofitable batch transport operations can lead to a giant tour that does not permit investigation of all aggregations due to the vehicle capacity.

The key point is that several transport operations between i and j should be aggregated (in the split procedure) if they are consecutive in the giant tour. This aggregation creates a single transport operation from several transport operations (with the quantity of resources equal to the sum of the transport operations), avoiding both loaded and unloaded transport operations.

For example, a unitary batch transport operation $\phi_{ij} = \{(i,j,*,1)(i,j,*,1)(i,j,*,1)\}$ can be aggregated in order to obtain one batch transport operation $\phi_{ij} = \{(i,j,*,3)\}$ composed of only one transport operation with three units of resources. The aggregation is possible only if there is a vehicle that can transport such a quantity of resources. An issue can arise in some cases, for example, if a batch transport operation $\phi_{ij} = \{(i,j,*,2)(i,j,*,2)\}$ is created from a batch flow transport operation $(i,j,*,\varphi_{ij})$ with $\varphi_{ij} = 4$. In this example, the split will not be able to aggregate the transport operation in order to obtain $(i,j,*,3)(i,j,*,1)$, for example, which could be the best aggregation/assignment.

Unitary batch transport operation

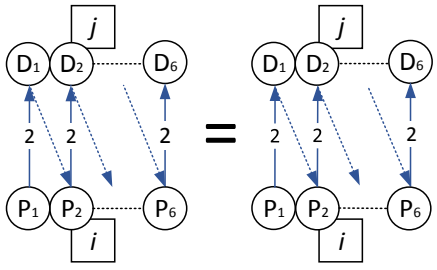
The unitary batch transport operation makes it possible to create a number of transport operations (equal to the flow value) and to assign all transport operations of the batch to all vehicles.

For any pickup operation on activity i and any delivery operation on activity j with a flow φ_{ij}, the problem consists in creating φ_{ij} transport operations $(i,j,*,1)$ to model all combinations of transport for all vehicles. This batch of transport operations can be written with a set of unitary transport operations $\phi_{ij} = \{(i,j,*,1), \dots, (i,j,*,1)\}$ with $Card(\phi_{ij}) = \varphi_{ij}$. With one unitary batch transport, the split procedure in the next step, can investigate all possible aggregations without any additional constraints. In spite of this advantage, such a situation is also unprofitable, first, for the split procedure that investigates all aggregations of transport operations (and assignment to vehicles) and, second, for the giant tour length.

Key features for an efficient batch transport operation

To favor the split procedure used next in the framework, it is of particular interest to provide a batch of transport operations with minimal length (minimal number of transport operations), leading to the same aggregations of transport operations as the unitary batch transport operations. Let us note $\phi^*_{ij} = \{(i,j,*,\varphi_1) \dots (i,j,*,\varphi_m)\}$ as the set of transport operation between i and j with the minimal number of transport operations.

Figure 10 illustrates an example composed of a heterogeneous fleet of three vehicles, with $c_1 = 5, c_2 = 3, c_3 = 2$ and a total amount of resources of $\varphi_{ij} = 12$ to transport from i to j.



Case 1

Case 1:
Let us consider a batch of transport operations where all transport operations are defined with two units of resources: $\phi_{ij} = \{(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)\}$.
Regardless of the vehicle (vehicle 2 or 3) used for ϕ_{ij}, the number of transport operations is exactly equal to 6 since no aggregation can be achieved (Fig. 10 - case 1).

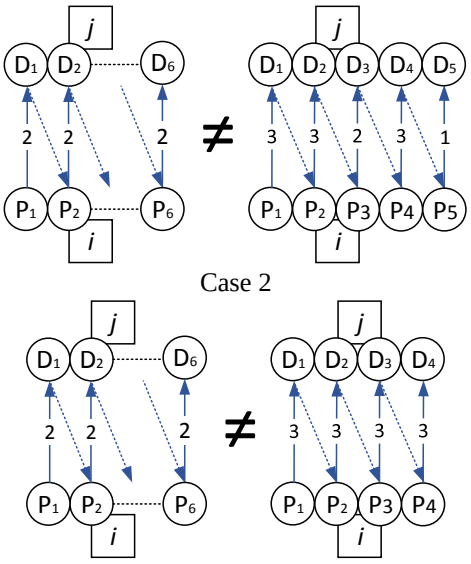


Fig. 10. Batch transport operation constraints on split with vehicles of capacity 2 and 3 (case 1 to 3)

Case 2:
If ϕ_{ij} is composed of transport operations with two or one units of resources, some transport operations can be aggregated. For example, with $\phi_{ij} = \{(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,1)(i,j,*,1)\}$, vehicle 3 requires six transport operations but vehicle 2 of capacity 3 can achieve only five transport operations, as illustrated in Fig. 10 (case 2).

Case 3:
A more promising set ϕ_{ij} is defined by $\phi_{ij} = \{(i,j,*,2)(i,j,*,1),(i,j,*,1)(i,j,*,2)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)\}$, where vehicle 2 can achieve only four transport operations (Fig. 10 - case 3).

Figure 10 proves that for the same pickup/delivery operation (pickup operation on activity i and delivery operation on activity j and a flow $\varphi_{ij} = 12$), it is possible to define different batch transport operations ϕ_{ij} (by defining the quantity of resources of each transport operation), leading to a different number of transport operations due to different possible aggregations. For example, $\phi_{ij} = \{(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,2)\}$ (case 3 of Figure 10), composed of eight transport operations, can be aggregated for vehicle 3 to define only four transport operations.

In all the cases introduced in Fig. 10, the choice of ϕ_{ij} does not impact the aggregation for vehicle 3, but case 3 for vehicle 2 is the most interesting case since it provides the minimal number of transport operations. Moreover, in case 3, ϕ_{ij} is composed of eight transport operations, which is less than 12 transport operations for the unitary batch transport operation. Another point can be highlighted by also taking vehicle 1 into account using case 3 (Fig. 11).

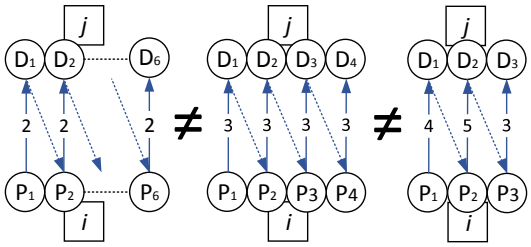


Fig. 11. Batch transport operation constraints on split with vehicles of capacity 2, 3 and 5

The problem of a batch transport definition must be addressed to create batch transport operations, with adapted solutions for all vehicles in the aggregation/assignment carried out in the split procedure.

Proposition for a batch transport definition

To create batch transport operations favoring the split procedure, the algorithm consists in investigating all cutoff points of all vehicles addressed separately with L_u, the ordered set of cutoff points of vehicle u. The set of cutoff points of one vehicle is defined by the sum of flow of each transport operation. Let us denote $L = \cup_u L_u$ as the ordered set of all cutoff points of all vehicles (without duplication).

For the first vehicle, the smallest batch transport operation is equal to $\phi_{ij} = \{(i,j,*,5)(i,j,*,5)(i,j,*,2)\}$, defining a set of cutoff points equal to $L_1 = (0,5,10,12)$. A similar remark holds for vehicle 2 considering $\phi_{ij} = \{(i,j,*,3)(i,j,*,3)(i,j,*,3)(i,j,*,3)\}$ which leads to the following set of cutoff points: $L_2 = (0,3,6,9,12)$. For the third vehicle with $\phi_{ij} = \{(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)(i,j,*,2)\}$, the set of cutoff points is: $L_3 = (0,2,4,6,8,10,12.)$. For this example, $L = (0,2,3,4,5,6,8,9,10,12)$.

Next, the transport operations of the unitary batch transport operation are iteratively scanned and gathered into a new transport operation $(i,j,*,\varphi_{ij})$. These transport operations are defined between two successive cutoff points of L and define ϕ_{ij}.

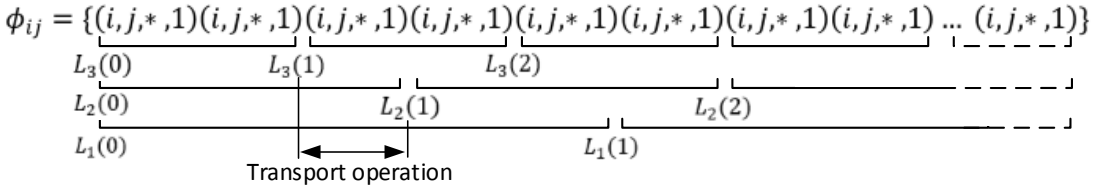


Fig. 12. Representation of the cutoff point on the unitary batch transport operation for the example with three vehicles, $c_1 = 5$, $c_2 = 3$, $c_3 = 2$ and a total amount of resources of $\varphi_{ij} = 12$

The best batch transport operation is built according to all the cutoff points. In this example, the best batch transport operation according to this definition is equal to:

$$\phi_{ij}^{*} = \{(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,1)(i,j,*,1)(i,j,*,1)(i,j,*,2)(i,j,*,1)(i,j,*,1)(i,j,*,1)\}.$$

4.6. Giant tour construction (third step of the evaluation)

The giant tour construction consists in scanning the set of batch flow transport operations with only one transport operation $(w(i),w(j),*,\varphi_{w(i)w(j)})$ in the increasing order of $w(i)$ and $w(j)$ and to schedule the batch transport operation $\phi_{w(i)w(j)}$ at the best possible location. The HGT procedure (Heuristic for Giant Tour) defines a depth-first-search strategy by a greedy insertion of each batch transport operation ϕ_{ij} at the best possible position. Let us note that the heuristic does not schedule transport operations one-by-one but, instead, the batch transport operations with all the transport operations that compose it. The construction is based on a `best_insertion` $(\lambda, cost)$ procedure that finds the best insertion position of the last scheduled batch transport operation with respect to the precedence and the resource constraints, calculating the minimal insertion cost equal to the empty transport between the last scheduled batch transport operation and the transport operation on the right of the insertion position.

Example:

Let us consider $w = [0,3,2,4,1,5,7,6,8,9]$ and the flow solution of Fig. 9. To build a giant tour, each batch transport operation associated with one flow is inserted at the best position. Some of the steps are detailed in Table 1.

Table 1
Steps to build a giant tour

	Batch to insert	Possible Giant Tours λ	Cost	Best
Step 1:	ϕ={(0,3,*,2)}	**(0,3,*,2)**	-	✓
Step 2:	ϕ={(0,2,*,2)}	(0,3,*,2)**(0,2,*,2)**	3	✓
		(0,2,*,2)(0,3,*,2)	4	
Step 3:	ϕ={(0,4,*,2)}	(0,3,*,2)(0,2,*,2)**(0,4,*,2)**	4	✓
		(0,3,*,2)**(0,4,*,2)**(0,2,*,2)	10	
		(0,4,*,2)(0,3,*,2)(0,2,*,2)	10	
Step 4:	ϕ={(3,1,*,2)}	(0,3,*,2)(0,2,*,2)(0,4,*,2)**(3,1,*,2)**	13	
		(0,3,*,2)(0,2,*,2)**(3,1,*,2)**(0,4,*,2)	3	✓
		(0,3,*,2)**(3,1,*,2)**(0,2,*,2)(0,4,*,2)	3	
…				
Step 14:	ϕ={(8,9,*,2),(8,9,*,1)}			
		(0,3,*,*,2)(0,2,*,*,2)(3,1,*,*,2)(0,4,*,2)(4,5,*,2) (5,6,*,2)(4,5,*,2)(5,8,*,1)(1,7,*,2)(7,8,*,2)(6,9,*,2) (7,9,*,1)**(8,9,*,2)(8,9,*,1)**	3	✓
		(0,3,*,2)(0,2,*,2)(3,1,*,2)(0,4,*,2)(4,5,*,2)(5,6,*,2) (4,5,*,2)(5,8,*,1)(1,7,*,2)(7,8,*,2)(6,9,*,2)**(8,9,*,2)** **(8,9,*,1)**(7,9,*,1)	4	
		(0,3,*,2)(0,2,*,2)(3,1,*,2)(0,4,*,2)(4,5,*,2)(5,6,*,2) (4,5,*,2)(5,8,*,1)(1,7,*,2)(7,8,*,2)**(8,9,*,2)** **(8,9,*,1)**(6,9,*,2)(7,9,*,1)	6	

In Step 4, the batch transport operation $\phi = \{(3,1,*,2)\}$ can be located on three positions but cannot be inserted at the first place of the sequence. The first transport operation $(0,3,*,2)$ defines a pickup of two units of flow at activity 0 and a delivery of two units at activity 3. Consequently, the transport $(3,1,*,2)$ cannot be scheduled before $(0,3,*,2)$ since $(3,1,*,2)$ defines a pickup at activity 3, which obviously required a previous delivery at activity 3.

In Step 14, the batch transport operation is composed of two transport operations that can be inserted on three positions, and the best insertion is located at the end of the giant tour.

4.7. Split algorithm (fourth step of the evaluation)

The split procedure, taken from routing, tackles only constraints taken from routing, including a limited number of vehicles, a limitation on hub capacity (see Duhamel et al. (2012) for a generic definition of split procedures). The main difference in the new split consists of the management of both the availability of vehicles and of activities. As highlighted in Duhamel et al. (2012), a label saved on a node models a state of the dynamic algorithm and must be defined by a cost and a full description of the system state. A split-based approach must tackle the following features:

- A label definition to model the system state;
- A propagation rule to create new labels;
- A dominance rule to save only non-dominated labels on the node.

A special feature of the split, dedicated to the RCPSPR, is the fact that the system state encompasses both the vehicle fleet state and the activity state:

- The vehicle fleet state is composed of three pieces of information for each vehicle: the position of the vehicle, the departure time and the arrival time. These data are required even if the fleet is homogeneous.
- The activity state is given by the earliest starting time of all the activities.

At each step of the split algorithm, the current system state is composed of the activity that has been scheduled but not achieved (see, for example, activity 5 in Fig. 13), activities previously scheduled, and vehicles at some specific location corresponding to their last loaded transport. Before achieving one transport operation (i,j,u,φ), the vehicle u has to be available and located on activity i, which may involve an unloaded transport. The cost of this unloaded transport depends of the current localization of the vehicle.

The split procedure consists of building an auxiliary acyclic graph H with $nt+1$ nodes numbered from 0 to nt (nt is the number of transport operations in the giant tour λ) where an arc from node i to j represents a subsequent μ_{ij} of transport operations from the giant tour with $\mu_{ij} = (\lambda(i+1),\ldots,\lambda(j))$, where $\lambda(i)$ is a transport operation. For the sake of convenience, $\lambda^{+}(i)$ denotes the pickup operation of $\lambda(i)$, $\lambda^{-}(i)$ the delivery operation of $\lambda(i)$, and $\lambda^{\varphi}(i)$ the flow associated with the transport operation $\lambda(i)$.

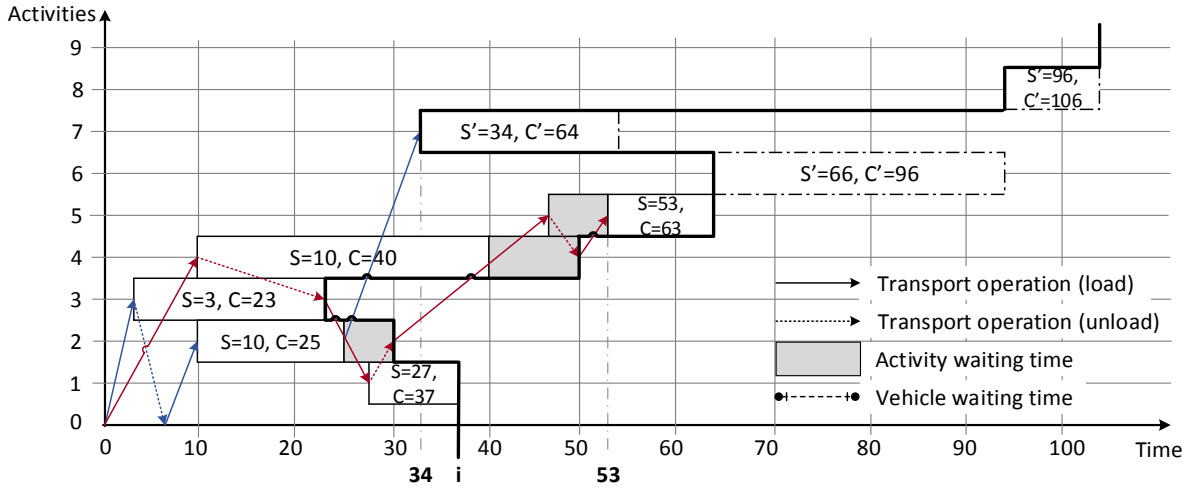


Fig. 13. System state during the Split at step i

The optimal splitting of λ (or quasi-optimal splitting in the event of extra constraints concerning the use of resources) corresponds to a min-cost path from node 0 to nt node in H. It can be computed using a Bellman-like algorithm taking the vehicle fleet into account, i.e., several labels have to be tackled per node. The principle is

endorsed by defining labels that model the current state of the global system, a propagation rule and a dominance between labels.

Label definition

To extract the vehicle trips from the giant tour, a set of non-dominated labels is stored on each node j of the graph H. A label represents a partial solution evaluation that tackles tasks $\lambda(0)$ to $\lambda(j)$. Let us denote L_i^j the i^{th} label on node j, which is composed of a $3v$-tuple V_i^j , a $n+1$-tuple S_i^j and of a couple (f_u, f_v):

- A $3v$-tuple V_i^j defining both vehicle times and position, $V_i^j = (B_{ij}^1, \dots, B_{ij}^v, A_{ij}^1, \dots, A_{ij}^v, P_{ij}^1, \dots, P_{ij}^v)$ where $P_{ij}^1, \dots, P_{ij}^v$ are the current position of each vehicle ($P_{ij}^u = p$ means that the position of the vehicle u is on activity p), $B_{ij}^1, \dots, B_{ij}^v$ give the earliest departure time of the vehicles, and $A_{ij}^1, \dots, A_{ij}^v$ are the arrival time of the vehicles.
- An $n+1$-tuple S_i^j defining the earliest starting time of each activity $S_i^j = (S_{ij}^0, \dots, S_{ij}^{n+1})$;
- A couple (f_u, f_v) of positions referring to the father label of L_i^j $i.e.$ L_i^j that has been generated using the label number f_u on node number f_v.

Label propagation

A propagation function defined by $f: \mathrm{L} \otimes \mathrm{T} \rightarrow \mathrm{R}$ permits the generation of a new label L_q^j from one label L_p^i and a trip evaluation γ depending of the subsequence μ_{ij} with a cost equal to transportation time from $\lambda^+(j)$ to $\lambda^-(j)$, i.e., $t_{\lambda^+(j),\lambda^-(j)}$. For a subsequent $\mu_{ij} = (\lambda(i+1), \dots, \lambda(j))$, the new label L_q^j for node j is defined from the label L_p^i considering the next update, using vehicle u:

$$B_{qj}^u = \max(S_{pi}^{\lambda^+(j)} + p_{\lambda^+(j)};\ B_{pi}^u + e_{P_{pi}^u,\lambda^+(j)})$$
$$B_{qj}^v = \max(B_{pi}^v;\ A_{pi}^v) \,/\, v \neq u$$
$$A_{qj}^u = B_{qj}^u + t_{\lambda^+(j),\lambda^-(j)}$$
$$P_{qj}^u = \lambda^-(j)$$
$$S_{qj}^{\lambda^-(j)} = \max(S_{pi}^{\lambda^-(j)};\ A_{qj}^u)$$

For the label L_p^i, all assignments to vehicles are investigated, leading to v labels. These final labels are included or not depending on the set of non-dominated labels previously stored on the node j.

Dominance rule

To keep only non-dominated labels on nodes, a dominance rule must be defined. Considering two labels L_p^i and P, L_p^i is defined as dominant as regards L_q^i ($L_q^i \ll L_p^i$) if the following two conditions hold:

Condition 1. All vehicles have the same location: $\forall u \in T, P_{ip}^u = P_{iq}^u$

Condition 2. $\forall u \in T, A_{ip}^u \leq A_{iq}^u$ and $\exists u \in T, A_{ip}^u < A_{iq}^u$

Let us note that the earliest starting time of activities is not relevant for the dominance rule since if all vehicle time availabilities of label L_p^i are less than or equal to vehicle availabilities on the label, then $L_q^i, \nexists k / S_{iq}^k < S_{ip}^k$.

Labels stored on nodes

If L_p^i is not dominant with regard to L_q^i ($L_q^i \not\ll L_p^i$), this does not imply that L_q^i is dominant with regard to L_p^i. If $L_q^i \not\ll L_p^i$ and $L_p^i \not\ll L_q^i$, then L_p^i cannot be compared to L_q^i. Thus, if $\exists k$ on node $i / L_p^i \ll L_k^i$, then L_p^i is not added to node i.

On the other hand, each label $L_k^i / L_k^i \ll L_p^i$ can be removed from node i. The dominance rule limits the number of labels stored at each node to a minor subset while maintaining split optimality. An additional time-saving approach consists in limiting the maximal number N_L of labels generated during the split process or per node. Such restrictions, in addition to the dominance rule, can strongly reduce the CPU time, but can yield a sub-optimal splitting solution with no real drawback on the global solution found at the end of the split, as reported by Boudia et al. (2007).

Figure 14 gives some steps of the split procedure applied on the giant tour of the example detailed in Table 1. The initial label on node 0 is $((0,0,0,0,0,0)(0,0,0,0,0,0,0,0,0,0))$ and it is propagated on the auxiliary graph with the arc between node 0 and 1 corresponding to the transport operation $(0,3,*,2)$ using the two possible vehicles. For

example, the transport operation can be carried out by vehicle 1 available at activity 0 on time 0. The duration of this transport operation is equal to 3, the propagation rule updates the arrival time of the vehicle to 3, the position is also equal to 3, and the earliest starting time of activity 3 has a value of 3. This process is iterated to obtain a set of eight labels on node 3 where each label represents a specific partial solution (see Fig. 14). With the dominance rule, only six of these eight labels are saved and the two underlined labels are deleted.

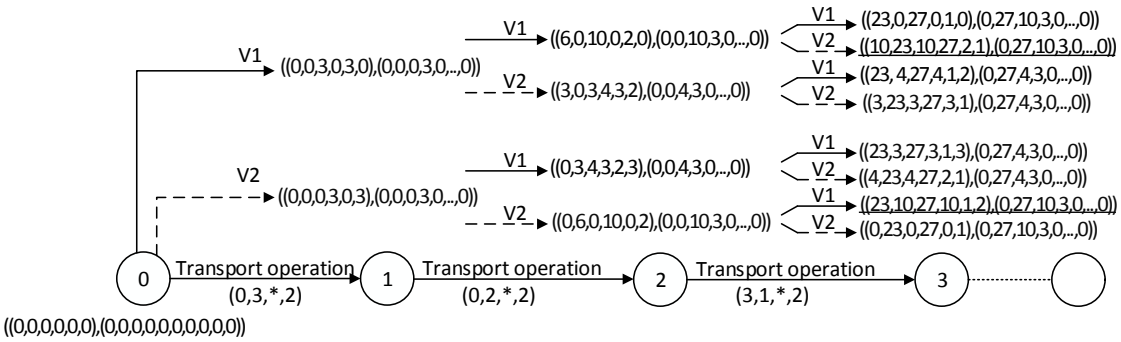


Fig. 14. Example of labels generated by the split in the auxiliary graph (data available at: http://fc.isima.fr/~vinot/Research/RCPSP&Routing.html)

```
   Procedure name: Split
1.  procedure Split
2.  input parameters
3.    λ: giant tour
4.    k: number of transport operation in the giant tour
5.  output parameters
6.    S: Solution graph
7.  global parameter
8.    n: number of activities
9.    v: number of vehicles
10.   NL: maximum number of labels on each node
11.  begin
12.    call InitGraph(S, λ, k, n, v, NL)
13.    i:=0
14.    while (i < k)
15.     j := i+1;  flow := 0
16.     while (CompareTransportOperation(λ, i+1, j)=1)
17.       flow := flow + λ^φ(j)
18.       for k := 1 to NBi do      //for each label on node i
19.         n_vehicle := 1
20.         while ((n_vehicle ≤ v) and (flow ≤ Cn_vehicle)) then
21.             P := call PropagateLabel(λ(j), flow, n_vehicle, L_k^i)
22.             if (NBj = 0) then call InsertLabel(P, j, S, NL)
23.             else    insert:=0
24.                     insert := call TestInsertLabel(P, j, S)
25.                     if (insert=1) then call InsertLabel(P, j, S, NL) endif
26.             endif
27.             n_vehicle := n_vehicle + 1
28.         endwhile
29.       endfor
30.         j := j + 1
31.     endwhile
32.     i := i + 1
33.   endwhile
34.   S := call Extract_trips ()  //save the best solution
35. end
```

Algorithm 2. Split procedure

The split is detailed in Algorithm 2 and uses low-level procedures to handle the labels:

- `InitGraph(S,λ,k,n,v,NL)` initializes all the local variables of the solution;
- `CompareTransportOperation(λ,i,j)` : the function returns 1 if the transport operation $\lambda(i+1)$ and $\lambda(j)$ have the same pickup and delivery operations, i.e., $\lambda^-(i+1) = \lambda^-(j)$ and $\lambda^+(i+1) = \lambda^+(j)$;
- `PropagateLabel(`$T_{(i,j,u,x)}$`,flow,v,L)` uses the propagation rule to generate the label P from L, knowing the transport operation $T_{(i,j,u,x)}$, the flow and the vehicle v used;

- `TestInsertLabel(P,j,S)` compares the label P to the labels on node j. If the label P dominates at least one label, the procedure returns 1 and the label dominated is deleted; otherwise, the procedure returns 0;
- `InsertLabel(P,j,S,N_L)` inserts the label P on node j in ascending order of the maximal value of the vehicle time availability. The number of labels on node j cannot exceed N_L;
- `Extract_trips()` validates the best final label and returns the solution with the set of trips.

The split algorithm (Algorithm 2) is composed of two parts: the initialization part where local variables are initialized (line 12) and a loop from lines 14 to 33 that iterates for the ordered set of customers defined by the giant tour λ. The second loop from lines 16 to 31 iterates and makes it possible to evaluate the partial sequence $(\lambda(i+1), \ldots, \lambda(j))$ on each label on node i with the loop from lines 18 to 29.

The third loop (from lines 20 to 28) iterates and makes it possible to calculate the partial sequence $(\lambda(i+1), \ldots, \lambda(j))$ using each vehicle with a sufficient capacity, leading to a label P (line 21), which is then inserted into the list of labels on node j, using the `InsertLabel` procedure. The best solution is saved at the end of the algorithm (line 34).

4.8. Scheduling and routing evaluation (fifth step of the evaluation)

The fully oriented disjunctive graph that models a solution is obtained, including both activity disjunctions and transport disjunctions and is then evaluated with a longest path algorithm to provide a semi-active solution.

4.9. Local search

Random initial solutions are obtained by random definition of activity sequence w limited to the space of non-level decreasing activity lists. In order to remove such restriction, neighborhoods can be obtained by the $Swap(u,v)$ operator, leading to a feasible move if there is no edge modeling a precedence constraint at indices $u+1$ to v. The whole space of the feasible activity sequence can be investigated.

To improve a RCPSPR solution, an efficient local search algorithm can be defined considering the critical path (Fig. 15 provides a graphical solution of the shortest path linked to the example of Fig. 10) with the introduction of blocks whose classification extends the scheduling blocks commonly used (Laarhoven et al., 1992; Matsuo et al., 1988; Dell'amico and Trubian, 1993; Nowicki and Smutnicki, 1996):

Block type 1: a transport operation and an activity;
Block type 2: two transport operations;
Block type 3: an activity and a transport operation;
Block type 4: two activities.

The local search investigates moves from the end of the critical path to the starting node and, depending on the block type encountered among the critical path, the actions are investigated on the activity sequence w only, thanks to the indirect representation of solutions. If one action leads to a lower cost solution, the critical path exploration is restarted at the end of the graph.

Block type 1 models a situation where the activity starting time depends on the transport operation (i,j,u,φ), which transports φ units of resource required by activity j. To obtain a neighborhood solution where this situation does not occur, the transport operation between the two activities must not be required. It can be obtained by a left shifting of the activity j into the activity sequence w to the left of activity i.

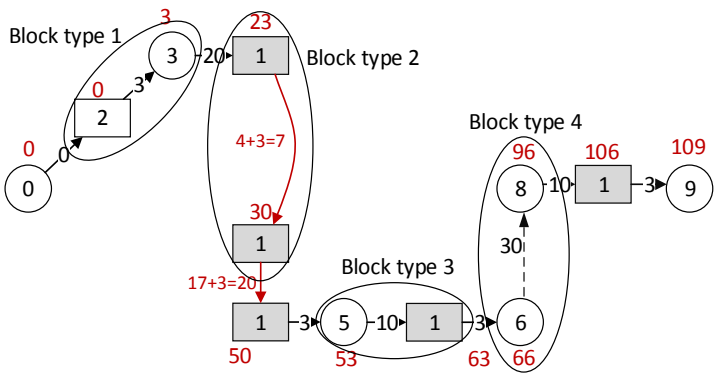


Fig. 15. Example of a critical path

Block type 2 concerns two consecutive transport operations (i, j, u, φ) and (k, p, v, φ'), including an unloaded transport operation. In this situation, the second transport operation (k, p, v, φ') has to wait to start for the end of the loaded transport operation (i, j, u, φ) and the end of the unloaded transport operations from j to k. It can be achieved by a left shifting of activity k to activity i.

Block type 3 models a situation that is quite identical to the situation of block type 1 and a similar approach holds.

Block type 4 between two activities occurs when an activity waits until the end of another activity to start, which is necessary due to a precedence constraint between them. To try to delete this arc from the critical path, the first activity has to be scheduled earlier. It can be achieved by a shifting of the activity in the activity sequence w with respect to the precedence constraint.

5 Numerical experiments

To the best of our knowledge, no instances dealing with this problem are available. Therefore, numerical experiments are based on the two sets of instances introduced below:

- The first one consists of the definition of a new set of small-scale instances with less than eight activities for which optimal resolution using CPLEX remains possible;
- The second one consists of the definition of medium-scale instances for about 30 activities not tractable by exact algorithms.

To favor fair future comparative studies, theses sets of instances are available at the following web page:

http://fc.isima.fr/~vinot/Research/RCPSP&Routing.html

The numerical experiments are carried out to meet the following requirements:

- To prove (using the small-scale instances) that the sequential resolution of the two sub-problems (the flow problem and then the routing/scheduling) leads to suboptimal solutions;
- To evaluate the performance of the approach by a careful comparison of the optimal solutions provided by CPLEX with a MILP and the solutions provided by the GRASP×ELS;
- To evaluate the GRASP×ELS convergence on medium-scale instances for future researchers.

The linear formulation used by CPLEX is provided in the appendix and additional numerical experiments have been achieved using a Constraint Programming model with IBM ILOG CPLEX Optimization Studio (the model is also provided in the appendix)

For each instance, five replications of the GRASP×ELS have been carried out. For the sake of convenience, all experiments are carried out with the same set of parameters introduced in Table 2 and the following notations are used for all of the results:

- Avg denotes an average value
- LB lower bound values (equal to the optimal solution of a RCPSP with transfer times)
- $LB(n_{tot})$ the minimal number of operations to be scheduled
- UB upper bound values
- BFS best solution found
- TT total CPU time in seconds to execute the algorithm (end of the iterations for the GRASP×ELS and time to close the gap for CPLEX)
- T* CPU time in seconds to obtain the BFS
- Nb. Opt Number of instances where the solution has the same value as the lower bound

Table 2
Parameters setting

Parameter	Definition	Value
np	Number of GRASP iterations	100
ne	Number of ELS iterations	25
nd	Number of neighborhood iterations	10
l	Number of local research iterations	10
NBmax	Maximum number of labels per node	20

All experiments are carried out on a single thread C program, using Visual Studio and Windows 7 as the operating system on a Dell Optiflex9020 with an Intel Core i7-4770 CPU 3.4 GHz and 16 Gb of RAM, which can be established at 2671 Mflops (see Dongarra et al. (2014)). The same set of parameters is used for all instances whatever the number of vehicles.

5.1. New set of instances

A new set of small-scale instances composed of 18 instances with six activities plus the start/end activity is introduced and encompasses two configurations for the location of activities: the first one with one uniform distribution of the activities and the second one with two clusters.

The resource requirement, the resource availability, the capacity of the vehicles and the ratio are given in Table 3. The minimal number of operations to be scheduled $LB(n_{tot})$ is given in column 2 in Table 3. These values represent the n activities to be scheduled, plus the minimal number of transport operations equal to $n + 1$ (with a sequential scheduling). All the details of the instances are available at the web page.

Table 3
Small scale instance characteristics

Instances	$LB(n_{tot})$	Location	Resources requirement	Resource availability	Vehicles capacity	Ratio
LMQV_U1	13	uniform	{4;10;3;3;8;4}	12	{12;12}	1.6
LMQV_U2	13	uniform	{4;10;3;3;8;4}	12	{12;12}	1.1
LMQV_U3	13	uniform	{4;10;3;3;8;4}	12	{12;12}	1.3
LMQV_U4	13	uniform	{2;8;7;8;8;5}	14	{6;5}	0.5
LMQV_U5	13	uniform	{2;8;7;8;8;5}	14	{6;5}	0.4
LMQV_U6	13	uniform	{2;8;7;8;8;5}	14	{6;5}	0.4
LMQV_U7	13	uniform	{5;3;8;2;3;1}	10	{4;2}	0.3
LMQV_U8	13	uniform	{5;3;8;2;3;1}	10	{4;2}	0.3
LMQV_U9	13	uniform	{5;3;8;2;3;1}	10	{4;2}	0.3
LMQV_C1	13	clusters	{7;7;5;7;5;4}	7	{7;7}	3.0
LMQV_C2	13	clusters	{7;7;5;7;5;4}	7	{7;7}	2.1
LMQV_C3	13	clusters	{7;7;5;7;5;4}	7	{7;7}	3.0
LMQV_C4	13	clusters	{7;1;2;6;2;6}	11	{6;5}	1.1
LMQV_C5	13	clusters	{7;1;2;6;2;6}	11	{6;5}	1.0
LMQV_C6	13	clusters	{7;1;2;6;2;6}	11	{6;5}	1.0
LMQV_C7	13	clusters	{4;10;4;6;3;4}	12	{4;2}	0.8
LMQV_C8	13	clusters	{4;10;4;6;3;4}	12	{4;2}	0.8
LMQV_C9	13	clusters	{4;10;4;6;3;4}	12	{4;2}	0.8

The medium-scale instance characteristics are introduced in Table 4 and encompass the quantification ratio, which is representative of the relative importance of the scheduling vs. the routing. The ratio is defined as the ratio between the average duration of an activity and the average duration of a transport operation per vehicle. To conclude, a ratio greater than 1 means that the scheduling processing time represents a greater amount of time than the routing, and a ratio lower than 1 implies the reverse.

Table 4
Medium-scale instance characteristics

Instances	$LB(n_{tot})$	Location	Resource requirement	Resource availability	Vehicle capacity	Ratio
LMQV_J30_U1	61	uniform	[1;10]	13	{13;13}	0.7
LMQV_J30_U2	61	uniform	[1;10]	14	{8;6}	0.3
LMQV_J30_U3	61	uniform	[1;10]	13	{13;9}	1.0
LMQV_J30_C1	61	clusters	[1;10]	15	{15;15}	0.8
LMQV_J30_C2	61	clusters	[1;10]	11	{7;5}	0.3
LMQV_J30_C3	61	clusters	[1;10]	12	{12;9}	0.9
LMQV_J30_CC1	61	clusters	[1;10]	13	{13;13}	0.6
LMQV_J30_CC2	61	clusters	[1;10]	14	{8;6}	0.3
LMQV_J30_CC3	61	clusters	[1;10]	13	{13;8}	1.1

5.2. Sequential resolution of the RCPSPR vs. the integrated approach

First, by using a linear flow formulation with CPLEX, an optimal solution of the RCPSP is provided. A second linear formulation then allows us to obtain the optimal routing/scheduling solution that is compliant with the optimal RCPSP flow solution. The sequential approach based on the successive execution of these two linear problems leads to one solution of the RCPSPR introduced in column 5 of Table 5.

Although the second linear model addresses only the routing/scheduling for a specific solution flow (RCPSP solution), the average computational time is about 18.1 seconds, proving that the joint resolution of scheduling/routings is a challenging problem. The solution of the sequential approach must be analyzed regarding the best solutions found for the GRASP×ELS in column 8.

The following remarks hold:

- All GRASP×ELS solutions are lower than the sequential solutions with a significant gap between the two approaches. For example, the GRASP×ELS provides a solution of 74 for the instance LMQV_U1, and the sequential resolution provides a solution of 95, which represents a gap of 28.4%. On average, the gap is about 23.6%.
- The sequential resolution defines a more time efficient approach since the total CPU time is about 18.1 seconds on average, six times lower than the average CPU time required by the integrated resolution with the GRASP×ELS.

Table 5
Sequential resolution vs. integrated resolution

		CPLEX MILP (sequential resolution)					GRASP×ELS (integrated resolution)		
Instances	$LB(n_{tot})$	RCPSP BFS	T (sec.)	RCPSPR BFS	T′ (sec.)	TT=T+T′ (sec.)	BFS	TT (sec.)	Gap (%)
LMQV_U1	13	19*	0.03	95	21.0	21.0	74	117.0	-28.4
LMQV_U2	13	27*	0.02	172	10.1	10.1	149	124.0	-15.4
LMQV_U3	13	38*	0.03	227	15.0	15.1	188	106.5	-20.7
LMQV_U4	13	21*	0.02	115	39.0	39.0	100	208.7	-15.0
LMQV_U5	13	38*	0.02	229	43.8	43.8	204	186.4	-12.3
LMQV_U6	13	42*	0.02	319	69.4	69.4	231	237.9	-38.1
LMQV_U7	13	8*	0.00	111	11.7	11.7	101	92.0	-9.9
LMQV_U8	13	15*	0.02	225	8.7	8.7	206	97.6	-9.2
LMQV_U9	13	16*	0.00	258	13.3	13.3	233	111.5	-10.7
LMQV_C1	13	45*	0.05	89	9.0	9.0	70	0.5	-27.1
LMQV_C2	13	65*	0.03	156	7.4	7.4	118	1.0	-32.2
LMQV_C3	13	90*	0.03	181	9.0	9.0	143	0.7	-26.6
LMQV_C4	13	12*	0.02	44	15.5	15.5	35	106.2	-25.7
LMQV_C5	13	32*	0.02	93	20.8	20.8	71	119.0	-31.0
LMQV_C6	13	24*	0.02	82	11.3	11.4	72	106.8	-13.9
LMQV_C7	13	18*	0.03	60	6.3	6.3	46	203.4	-30.4
LMQV_C8	13	35*	0.05	135	7.3	7.3	96	140.1	-40.6
LMQV_C9	13	36*	0.02	138	7.7	7.7	100	145.9	-38.0
Avg.			0.02		18.1	18.1		117.0	-23.6

To conclude, the results comply with the theoretical considerations, proving that sequential approaches lead to poor quality solutions. It should be noted that these results confirm the assertion of Section 4, proving that a RCPSP quality solution is not a guarantee of success for the RCPSPR.

5.3. Integrated approaches: GRASP×ELS solution vs. CPLEX optimal RCPSPR resolution

The linear formulation of the RCPSPR (Lacomme et al., 2017), encompasses a formulation with a large number of constraints and variables: for the instance LMQV_U1, for example, the linear formulation is composed of 54,186 variables and 140,974 constraints, including binary variables to model the disjunction of activities. Only 14 instances are solved to optimality in less than one day of computation time (Table 6). In the optimal solution of instance LMQV_C9, the linear program has to deal with six activities and 26 transport operations, representing 32 operations to be scheduled. The notation t.l. is used in tables for computational time exceeding 48 hours.

Table 6
RCPSPR optimal resolution

Instances	$LB(n_{tot})$	r	CPLEX MILP (integrated approach)			
			nc	nr	Optimal Solution	TT (sec.)
LMQV_U1	13	1.6	54186	140974	74	18 792
LMQV_U2	13	1.1	54186	140974	144	17 892
LMQV_U3	13	1.3	54186	140974	188	23 976
LMQV_U4	13	0.5	53999	140584	74	t.l.
LMQV_U5	13	0.4	53999	140584	192	71 172
LMQV_U6	13	0.4	53999	140584	228	t.l.
LMQV_U7	13	0.3	13421	34831	97	14 580
LMQV_U8	13	0.3	13421	34831	197	18 864
LMQV_U9	13	0.3	13421	34831	218	56 988
LMQV_C1	13	3.0	84523	220446	70	4 104
LMQV_C2	13	2.1	84523	220446	118	5 364
LMQV_C3	13	3.0	84523	220446	143	5 724
LMQV_C4	13	1.1	21072	54979	33	1 512
LMQV_C5	13	1.0	21072	54979	68	1 224
LMQV_C6	13	1.0	21072	54979	72	1 296
LMQV_C7	13	0.8	27848	72442	44	54 576
LMQV_C8	13	0.8	27848	72442	90	t.l.
LMQV_C9	13	0.8	27848	72442	94	t.l.
		1.1	42508	110709		>86 400

t.l. : time greater than 48 hours

Table 7 provides a comparative study between the best found solutions with CPLEX MILP and the GRASP×ELS resolution where both methods solve the RCPSPR using an integrated approach. Best solutions proved to be optimal are tagged with *.

Table 7
GRASP×ELS resolution

Instances	$LB(n_{tot})$	LB	CPLEX MILP (integrated approach)		GRASP×ELS (integrated approach)			
			BFS	TT(sec.)	BFS	T* (sec.)	TT (sec.)	Gap (%)
LMQV_U1	13	57	74*	18 792	74*	58.6	117.0	0.0
LMQV_U2	13	112	144*	17 892	149	65.9	124.0	3.5
LMQV_U3	13	148	188*	23 976	188*	64.8	106.5	0.0
LMQV_U4	13	79	95*	t.l.	100	131.2	208.7	5.3
LMQV_U5	13	157	192*	71 172	204	140.4	186.4	6.3
LMQV_U6	13	178	228*	t.l.	231	0.0	237.9	1.3
LMQV_U7	13	48	97*	14 580	101	15.1	92.0	4.1
LMQV_U8	13	96	197*	18 864	206	18.1	97.6	4.6
LMQV_U9	13	114	218*	56 988	233	0.1	111.5	6.9
LMQV_C1	13	70	70*	4 104	70*	0.0	0.5	0.0
LMQV_C2	13	118	118*	5 364	118*	0.3	1.0	0.0
LMQV_C3	13	143	143*	5 724	143*	0.0	0.7	0.0
LMQV_C4	13	25	33*	1 512	35	2.7	106.2	6.1
LMQV_C5	13	50	68*	1 224	71	2.7	119.0	4.4
LMQV_C6	13	51	72*	1 296	72*	62.8	106.8	0.0
LMQV_C7	13	33	44*	54 576	46	0.2	203.4	4.5
LMQV_C8	13	66	90*	t.l.	96	91.9	140.1	6.7
LMQV_C9	13	68	94*	t.l.	100	0.2	145.9	6.4
Avg.				>86 400		36.4	117.0	3.3

*: optimal solution

The average computational time of the GRASP×ELS is about 36.4 seconds and the average deviation between the GRASP×ELS solution and the CPLEX solution is about 3.3%, proving that in some instances (for example, on LMQV_U3), the GRASP×ELS has the capacity to find the optimal solution (solution cost of 188) in 64.8 s, whereas CPLEX requires more than 23 976 seconds (six hours) of computational time.

Figure 16 provides the fully oriented disjunctive graph that models the optimal LMQV_C4 solution. This graph encompasses the disjunctions between activities, the vehicle assignment to transport operations and the disjunctions between the transport operations. Integer values in front of nodes are the earliest starting time of operations.

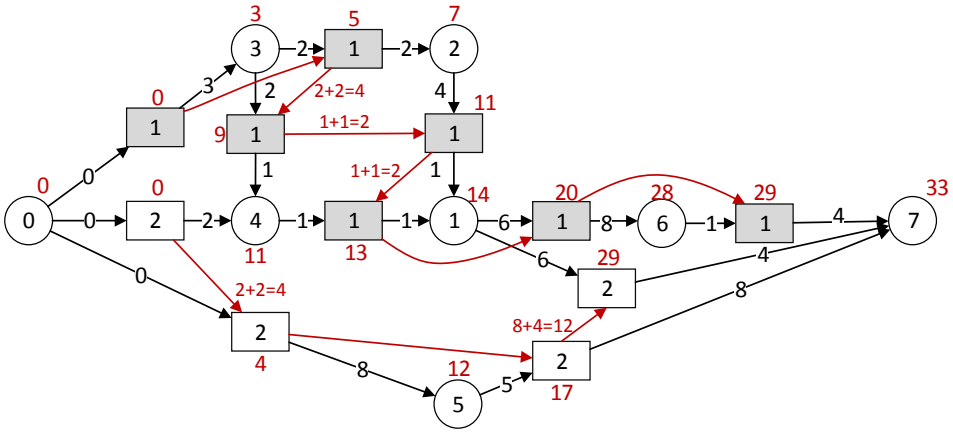


Fig. 16. Example of solution for LMQV_C4 (CPLEX solution)

The set of instances is extended to tackle up to five vehicles (all vehicles capacity are defined on the web companion page) to evaluate the impact of routing with regard to the scheduling solutions (Table 8). It can be observed that by increasing the number of vehicles from three to four, the number of solutions proved to optimality increases from 6 to 15 over the 18 instances.

Table 8
GRASP×ELS resolution from three to five vehicles on small scale instances

			GRASP×ELS (3 vehicles)			GRASP×ELS (4 vehicles)			GRASP×ELS (5 vehicles)		
Instances	$LB(n_{tot})$	LB	BFS	T* (s.)	TT (s.)	BFS	T* (s.)	TT (s.)	BFS	T* (s.)	TT (s.)
LMQV_U1	13	57	62	0.1	67.0	57	1.3	78.1	57	1.3	75.3
LMQV_U2	13	112	118	0.6	51.5	113	1.1	75.2	112	1.5	95.8
LMQV_U3	13	148	159	1.0	62.2	148	10.7	61.7	148	1.6	38.3
LMQV_U4	13	**79**	**79**	3.7	53.7	79	0.3	79.9	79	0.6	82.6
LMQV_U5	13	157	157	10.7	76.0	157	0.5	74.6	157	0.6	85.9
LMQV_U6	13	178	178	48.1	125.1	178	0.5	95.2	178	0.5	95.7
LMQV_U7	13	48	58	0.4	52.6	48	14.0	70.0	48	0.4	65.9
LMQV_U8	13	**96**	118	4.6	47.1	97	13.1	61.4	**97**	0.2	58.4
LMQV_U9	13	114	147	15.1	63.1	114	2.2	47.5	114	9.5	51.2
LMQV_C1	13	70	70	0.4	2.7	70	2.0	4.7	70	2.6	5.5
LMQV_C2	13	118	118	0.0	3.5	118	0.0	4.3	118	3.2	7.7
LMQV_C3	13	143	143	0.9	4.5	143	0.0	2.5	143	0.0	5.7
LMQV_C4	13	25	29	0.4	39.3	25	0.7	40.5	25	0.0	50.6
LMQV_C5	13	50	59	0.9	49.0	50	1.1	40.8	50	1.5	58.2
LMQV_C6	13	51	61	0.1	67.0	52	0.9	40.8	51	0.5	59.5
LMQV_C7	13	33	38	0.6	51.5	33	10.2	70.4	33	14.3	111.5
LMQV_C8	13	66	73	1.0	62.2	66	64.7	146.7	66	7.5	80.6
LMQV_C9	13	68	72	3.7	53.7	68	1.1	81.6	68	115.3	232.9
Avg.				6.2	49.8		7.9	55.8		11.1	69.2
Nb. Opt			6/18			15/18			17/18		

After intensive numerical experiments, it has been proved that a fleet of five vehicles is sufficient to reach the lower bound of all instances except instance LMQV_U8 where the best solution found values 97 and the lower boud is about 96. The set of parameters should be tuned to the number of vehicles since the number of solutions strongly depends on the vehicles.

One can noticed that a sequential resolution of the RCPSPR, starting from a flow solution provided by the optimal solution of the RCPSP with tranfer times, does not ensure a good quality solution for any number of vehicles. For example, in the instance LMQV_U4, the optimal resolution of the RCPSPR, starting from the flow leading to the optimal solution of the RCPSP with tranfer times, does not make it possible to reach the lower bound value equal to 79. With such a sequential approach, the best solutions are respectivelly equals to 125, 88 and 79, for two, three and four vehicles. Similar remarks hold for a waste majority of instances and push us into considering that a sequential resolution based on the RCPSP with tranfer times is not a efficient approach.

5.4. GRASP×ELS solution vs. optimal RCPSPR resolution: medium-scale instances

A new set of medium-scale instances composed of nine instances with 32 activities including the dummy ones is used to evaluate the GRASP×ELS performances. The results of Table 9 are not compared to the MILP model since CPLEX fails to find a solution after five days.

Table 9
GRASP×ELS resolution for medium-scale instances

Instances	$LB(n_{tot})$	r	GRASP×ELS		
			BFS	T* (sec.)	TT (sec.)
LMQV_J30_U1	61	0.7	175	37.5	213.2
LMQV_J30_U2	61	0.3	197	338.9	576.1
LMQV_J30_U3	61	1.0	107	27.7	244.6
LMQV_J30_C1	61	0.8	175	182.8	524.9
LMQV_J30_C2	61	0.3	177	51.2	172.4
LMQV_J30_C3	61	0.9	115	100.2	241.8
LMQV_J30_CC1	61	0.6	188	62.8	465.5
LMQV_J30_CC2	61	0.3	195	29.6	397.1
LMQV_J30_CC3	61	1.1	124	27.2	251.9
Avg.				95.3	343.0

The average computational time of the GRASP×ELS is about 343.0 seconds. All the details of the instances are available at the web page.

6 Concluding remarks

The integration of scheduling and routing is the key feature of supply chains for the proper coordination of these two functions in order to obtain quality results. This paper addresses the resolution of the RCPSPR, extending the RCPSP with the addition of routing using a GRASP×ELS metaheuristic. The framework we promote takes advantage of an indirect representation of the solutions using a split-based approach, and of different key features to favor the search space alternation with the flow, the giant tour and RCPSPR solutions. The framework efficiency takes advantage of a disjunctive graph with the integration of scheduling and routing constraints. The framework efficiency is proved by an intensive numerical experiments with CPLEX optimal solutions on a set of small- and medium-scale instances. The approach promoted takes advantages of Artigues et al.'s flow formulation and this model is relevant when resources are not allowed to make trips and only direct resource transfers are permitted. New research could be directed into RCPSP flow formulation that could be used to create trip with indirect resource transfer.

Our research is now directed toward a resolution with several resources and multi-objective functions where several criteria could be introduced for a quality of service that could be defined as the waiting time for both resources (the resources transported and the vehicles). This second criterion should be relevant for the quality of service when the resources provided for activities are linked to perishability constraints, for example, or when inventory costs have to be taken into account. Future perspectives focus on strengthen both MILP and CP formulations using, but not limited to, the definition of efficient lower bounds and upper bounds and additional constraints.

Acknowledgements: This work was carried out and funded within the framework of the ATHENA project (Reference: ANR-13-BS02-0006) and the Labex MS2T. It was supported by the French government through the "Investments for the future" program managed by the French National Research Agency (Reference: ANR-11-IDEX-0004-02).

References

Adnen, E.A. & Mohsen, E. (2016). An efficient new heuristic for the hoist scheduling problem. Computers & Operations Research, 67, 184-192.

Afsar, H.M., Lacomme, P., Ren, L., Prodhon, C. & Vigo, D. (2016). Resolution of a Job-Shop problem with transportation constraints: a master/slave approach. IFAC-PapersOnLine, 49(12), 898-903.

Ahuja, R.V., Magnanti, T.L. & Orlin, J.B. (1993). Network Flows: Theory/Applications, Prentice hall, N.J.

Alvarez-Valdés, R. & Tamarit, J.M. (1993). The project scheduling prolyhedron: Dimension, facets and lifting theorems. EJOR, 67, 204-220.

Artigues, C. & Roubellat, F. (2000). A polynomial insertion algorithm in RCPSP with cumulative constraints and multiple nodes. EJOR 127(2), 297- 316.

Artigues, C., Michelon, P. & Reusser, S. (2003). Insertion for static/dyn. RCPSP. EJOR 149, 249-67.

Beasley, J.E. (1983). Route-first cluster-second methods for vehicle routing. Omega, 11, 403-408.

Blazewicz, J., Lenstra, J.K. & Rinooy Kan, A.H.G. (1983). Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics, 5, 11-24.

Boudia, M., Prins, C. & Reghioui, M. (2007). An effective memetic algorithm with population management for the split delivery vehicle routing problem. In Hybrid Metaheuristics, Springer Berlin Heidelberg, 16-30.

Brucker, P., Knust, S., Roper, D. & Zinde, Y. (2000). Scheduling UET task system with concurrency on two parallel identical processors. Mathematical Methods of Operation Research, 53, 369-387.

Brucker, P., Drexl, A., Mohring, R., Neumann, K. & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. EJOR, 112(1), 3-41.

Caumond, A., Lacomme, P., Moukrim, A. and Tchernev, N. (2009). An MILP for scheduling problems in a FMS with one vehicle. EJOR, 199(3), 706-722.

Chassaing, M. (2015). Problèmes de transport à la demande avec prise en compte de la qualité de service. Doctoral dissertation. Blaise Pascal University, Clermont-Ferrand, N° d'ordre: D. U: 2631.
Cheng, R., Gen, M. & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms – I representation. Computers and industrial engineering, 30, 983–997.
Chen, Z.L. (2010). Integrated Production and Outbound Distribution Scheduling: Review and Extensions. Operations Research, 58(1), 130-148.
Chtourou, S., Manier, M.-A. & Loukil, T. (2013). A hybrid algorithm for the cyclic hoist scheduling problem with two transportation resources. Computers & Industrial Engineering, 65(3), 426-437.
Cordeau, J.-F. & Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a ride problem. Transp. Res. Part B Methodol, 37, 579–594.
Dauzère-Pérès, S. & Lasserre, J.B. (1995). A new mixed-integer formulation of the flow-shop sequencing problem. In: 2nd Workshop on models and algorithms for planning and scheduling problems, Wernigerode, Germany.
Dell'amico, M. & Trubian, M. (1993). Applying tabu-search to the job-shop scheduling problem. Annals of Operations Research, 41, 231-252.
Demeulemeester, E. & Herroelen, W. (2002). Project scheduling – A research and book International Series. Operations Research and Management Science, Kluwer Academic Publishers Boston.
Dongarra, J. (2014). Performance of various computers using standard linear equations software. Report CS-89-85, University of Manchester.
Duhamel, C., Lacomme, P. & Prodhon, C. (2012). A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems. Eng. Appl. Artif. Intell, 25(2), 345-358.
Duhamel, C., Lacomme, P. & Prodhon, C. (2011). Efficient frameworks for greedy split and new depth first search split procedures for routing problems. Computers & Operations Research, 38(4), 723-739.
Feo, T.A. & Resende, M.G.C. (1995). Greedy Randomized Adaptive Search Procedures. J. Glob. Optim, 6(2), 109-133.
Firat, M. & Woeginger, G.J. (2011). Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Operations Research Letters, 39, 32–35.
Hartmann, S. & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research, 207(1), 1-14.
Herroelen, W., De Reyck, B. & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. Computers and Operations Research, 25(4), 279-302.
Honglin, Q., Sujing, W. & Qiang, Xu. (2016). A new method of cyclic hoist scheduling for multi-recipe and multi-stage material handling processes. Computers & Chemical Engineering, 90, 171-187.
Knust, S. (1999). Shop-Scheduling Problems with Transportation. Ph.D. thesis. Fachbereich Mathematik/ Informatik, Universität Osnabrück.
Kolisch, R. & Padman, R. (2001). An integrated survey of deterministic project scheduling. OMEGA, 29, 249-272.
Krüger, D. & Scholl, A. (2009). A heuristic solution framework for the resource constrained (multi-) project scheduling problem with sequence-dependent transfer time. EJOR, 197(2), 492-508.
Kreter S., Rieck J. & Zimmermann JA. (2016). Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars. European Journal of Operational Research. 251:387-403.
Laarhoven, V.P.J.M, Aarts, E.H.L. & Lenstra, J.K. (1992). Job shop scheduling by Simulated Annealing. Operation Research, 40(1), 113-125.
Lacomme, P., Larabi, M. & Tchernev, N. (2007). A disjunctive graph for the job-shop with several robots. In: MISTA conference, Paris, France.
Lacomme, P., Larabi, M. & Tchernev, N. (2010). Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Economics, 143(1), 24-34.
Lacomme, P., Prins, C. & Ramdane-Cherif, W. (2001). Competitive genetic algorithms for the Capacitated Arc Routing Problem and its extensions. Lecture Notes in Computer Science, 2037, 473-483.
Lacomme, P., Prins, C. & Ramdane-Chérif, W. (2004). Competitive memetic algorithms for arc routing problems. Annals of Operations Research, 131,159-185.
Lacomme, P., Moukrim, A., Quilliot, A. & Vinot, M. (2017). Formalisation linéaire d'un problème de RCPSP avec transport de ressources, 18ème congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF), Metz, France.
Matsuo, H., Suh, C.J. & Sullivan, R.S. (1988). A controlled search simulated annealing method for the general job-shop scheduling problem. Working Paper, 03-04- 88, Graduate School of Business, University of Texas at Austin, Austin, Texas, USA.
Nowicki, E. & Smutnicki, C. (1996). A fast taboo search algorithm for the job-shop problem. Management Science, 42(6), 797-813.
Poppenborg, J. & Knust, S. (2016). A flow-based tabu search algorithm for the RCPSP with transfer times. OR spectrum, 38(2), 305-334.
Prins, C. (2009). A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem. Bio-inspired Algorithms for the Vehicle Routing Problem, 161, 35-53.
Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research, 31(12), 1985-2002.
Prins, C., Lacomme, P. & Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. Transportation Research Part C: Emerging Technologies, 40, 179-200.
Pritsker, A. & Watters, L. (1968). A zero-one programming approach to scheduling with limited resources. The RAND Corporation, RM-5561- PR.
Pritsker, A., Watters, L. & Wolfe, P. (1969). Multi-project scheduling with limited resources: a zero-one programming approach. Manage. Sci., 16:93-108.
Quilliot, A. & Toussaint, H. (2012). Resource Constrained Project Scheduling with Transportation Delays, IFAC Proceedings Volumes, 45(6), 1481-1486.
Roy, B. & Sussmann, B. (1964). Les problèmes d'ordonnancement avec contraintes disjonctives. Note DS no. 9 bis, SEMA, Paris.
Weiss, I., & Schwindt, C. (2016). The resource transfer Problem: Modeling and solving integrated scheduling and routing problems. In Industrial Engineering and Engineering Management (IEEM), IEEE International Conference, 755-759).
Weglarz, J. (1999). Project Scheduling. Recent Models, Algorithms and Applications. Kluwer Academic Publishers.
Wolf, S. & Merz, P. (2007). Evolutionary local search for the super-peer selection problem and the p-hub median problem. Lecture notes in computer science, 4771, 1-15.
Zhang, Q., Manier, H. & Manier, M.-A. (2012). A Modified Disjunctive Graph for Job-Shop Scheduling Problems with Bounded Processing Times and Transportation Constraints. IFAC Proceedings, 45(6), 1377-1382.
Zhang, Q., Manier, H. & Manier, M.-A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computers & Operations Research, 39(7), 1713-1723.

Appendix

A.1. MILP formulation for the RCPSPR

Data of the problem

- p_i processing time of activity i
- b_i number of resource required for activity i
- t_{ij} transfer time from activity i to activity i
- V set of activities to schedule
- E set of couple of activities linked by a precedence constraints
- T set of vehicle
- T_i vehicle i
- C_i capacity of vehicle i

Variables

- φ_{ijk} number of resource k transferred from i to j
- z_{ij} binary variable that value 1 if activity i is scheduled before activity j
- S_i starting time of activity i
- y_{ijut} amount of resource transported from activity i to j by vehicle u during the t^{th} transport operation
- x_{ijut} binary variable such $x_{ijut} = 1$ if vehicle u make the transport of resource from activity i to j during the t^{th} transport operation.
- a^u_{ijtpqv} binary variable with $a^u_{ijtpqv} = 1$ if transport operation number t of vehicle u from activity i to j is schedule before the transport operation number v of vehicle u from activity p to q

The constraints numbered 1 to 9 in $P1$ are the classical RCPSP constraints (scheduling constraint) and constraints numbered 10 to 25 in $P2$ are the constraints tackling the transport.

$$P1: \begin{cases} & Minimize\ S_{n+1} & (1) \\ \forall\,(i,j) \in E & z_{ij} = 1 & (2) \\ \forall\,(i,j) \in V^2 & S_j \geq S_i + p_i + M(z_{ij} - 1) & (3) \\ \forall\,(i,j) \in V^2 & \varphi_{ij} \leq min(b_i, b_j) z_{ij} & (4) \\ \forall\, j \in V & \sum_{i \in V} \varphi_{ij} = b_j & (5) \\ \forall\, i \in V & \sum_{j \in V} \varphi_{ij} = b_i & (6) \\ \forall\,(i,j) \in V^2 & z_{ij} \in \{0,1\}, z_{ii} = 0 & (7) \\ \forall\, i \in V & S_i \in \mathbb{N} & (8) \\ \forall\,(i,j) \in V^2 & \varphi_{ij} \in \mathbb{N}, \varphi_{ii} = 0 & (9) \end{cases}$$

$$P2:\begin{cases}
\forall (i,j) \in V^2 & \sum_{u\in T}\sum_{t\in X_{ij}} y_{ijut} = \varphi_{ij} & (10)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & y_{ijut} \le min(b_i, b_j, C_u)x_{ijut} & (11)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & x_{ijut} \le y_{ijut} & (12)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & A_{ijut} \ge B_{ijut} + t_{ij} + (x_{ijut} - 1).H & (13)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & S_j \ge A_{ijut} + (x_{ijut} - 1).H & (14)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & B_{ijut} \ge S_i + p_i + (x_{ijut} - 1).H & (15)\\
\forall (i,j,p,q) \in V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a_{ijtpqv}^u + a_{pqvijt}^u \le x_{ijut} & (16)\\
\forall (i,j,p,q) \in V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a_{ijtpqv}^u + a_{pqvijt}^u \le x_{pquv} & (17)\\
\forall (i,j,p,q) \in V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a_{ijtpqv}^u + a_{pqvijt}^u \ge 1 - (2 - x_{ijut} - x_{pquv}).H & (18)\\
\forall (i,j,p,q) \in V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & B_{pquv} \ge A_{ijut} + t_{jp} + (a_{ijtpqv}^u - 1).H & (19)\\
\forall (i,j) \in V^2,\ \forall u \in T, \forall t \in [\![1, min(b_i, b_j) - 1]\!] & y_{ijut+1} \le y_{ijut} & (20)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & y_{ijut} \in \mathbb{N} & (21)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & x_{ijut} \in \{0,1\} & (22)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & A_{ijut} \in \mathbb{N} & (23)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & B_{ijut} \in \mathbb{N} & (24)\\
\forall (i,j,p,q) \in V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a_{ijtpqv}^u \in \{0,1\} & (25)
\end{cases}$$

A.2. CP formulation for the RCPSPR

The CP model was formulated from the MILP and both CP data and CP decision variables have been introduced considering the same trend as the MILP, in order to favor readability first, and to facilitate second a better knowledge and understanding for researchers coming from the scheduling community and who are more experienced in MILP than CP. A CP model resembles an integer programming model in terms of syntax. It contains decision variables with their domains, a set of constraints, and possibly, an objective function. However, the CP modeling paradigm is much more expressive. In fact, the language is a superset of the integer linear programming modeling language. In addition to equality and inequality constraints between linear mathematical expressions, a CP model can contain non-linear expressions, logical expressions, use decision variables as indices to other vectors of decision variables, and include global constraints that capture a relationship between large sets of decision variables.

Disjunctive formulation of the CP

$$P1:\begin{cases}
 & Minimize\ S_{n+1} & (1)\\
\forall (i,j) \in E & z_{ij} = 1 & (2)\\
\forall (i,j) \in V^2 & z_{ij} = 1 \Longrightarrow S_j \ge S_i + p_i & (3)\\
\forall (i,j) \in V^2 & \varphi_{ij} \le min(b_i, b_j) z_{ij} & (4)\\
\forall j \in V & \sum_{i\in V} \varphi_{ij} = b_j & (5)\\
\forall i \in V & \sum_{j\in V} \varphi_{ij} = b_i & (6)\\
\forall (i,j) \in V^2 & z_{ij} \in \{0,1\},\ z_{ii} = 0 & (7)\\
\forall i \in V & S_i \in \mathbb{N} & (8)\\
\forall (i,j) \in V^2 & \varphi_{ij} \in \mathbb{N}, \varphi_{ii} = 0 & (9)
\end{cases}$$

$$P2:\begin{cases}
\forall (i,j) \in V^2 & \sum_{u\in T}\sum_{t\in X_{ij}} y_{ijut} = \varphi_{ij} & (10)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & y_{ijut} \le min(b_i, b_j, C_u)x_{ijut} & (11)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & y_{ijut} = 0 \Longrightarrow\ x_{ijut} = 0\ \wedge\ A_{ijut} = 0\ \wedge\ B_{ijut} = 0 & (12)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & x_{ijut} = 1 \Longrightarrow A_{ijut} \ge B_{ijut} + t_{ij} & (13)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & x_{ijut} = 1 \Longrightarrow S_j \ge A_{ijut} & (14)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & x_{ijut} = 1 \Longrightarrow B_{ijut} \ge S_i + p_i & (15)\\
\forall (i,j,p,q)\ \in\ V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} + a^u_{pqvijt} \le x_{ijut} & (16)\\
\forall (i,j,p,q)\ \in\ V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} + a^u_{pqvijt} \le x_{pquv} & (17)\\
\forall (i,j,p,q)\ \in\ V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & \left(x_{ijut} = 1\ \wedge x_{pquv} = 1\right) \Longrightarrow a^u_{ijtpqv} + a^u_{pqvijt} \ge 1 & (18)\\
\forall (i,j,p,q)\ \in\ V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} = 1\ \Longrightarrow B_{pquv} \ge A_{ijut} + t_{jp} & (19)\\
\forall (i,j) \in V^2,\ \forall u \in\ T, \forall t \in [\![1, min(b_i, b_j) - 1]\!] & y_{ijut+1} \le y_{ijut} & (20)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & y_{ijut} \in \{0, B\} & (21)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & x_{ijut} \in \{0,1\} & (22)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & A_{ijut} \in \mathbb{N} & (23)\\
\forall (i,j) \in V^2,\ \forall u \in T,\ \forall t \in X_{ij} & B_{ijut} \in \mathbb{N} & (24)\\
\forall (i,j,p,q)\ \in\ V^4, \forall u \in T, \forall (t,v) \in X_{ij}^2 & a^u_{ijtpqv} \in \{0,1\} & (25)
\end{cases}$$

<u>Cumulative formulation of the CP for the scheduling part</u>

In the cumulative formulation, the activities are represented by interval variables. An interval variable represents an interval of time during which an activity is performed:

```
interval act[i in 0..n+1] size pi
```

The cumulative constraint to define f enforces that at each point in time, the cumulated resource demands of the set of activities that require the resource at this time, does not exceed a given limit.

```
cumulFunction f = sum(i in 1..n) pulse(act[i],bi)
```

$$P1':\begin{cases}
 & Minimize\ \text{endOf}(act[n+1]); & (1)\\
 & f\ \le\ B; & (2)\\
\forall (i,j) \in V^2 & z_{ij} = 1 \Longrightarrow \text{endBeforeStart}(act[j], act[i]); & (3)\\
\forall (i,j) \in V^2 & \varphi_{ij} \le min(b_i, b_j)z_{ij} & (4)\\
\forall j \in V & \sum_{i\in V} \varphi_{ij} = b_j & (5)\\
\forall i \in V & \sum_{j\in V} \varphi_{ij} = b_i & (6)\\
\forall (i,j) \in V^2 & z_{ij} \in \{0,1\},\ z_{ii} = 0 & (7)\\
\forall i \in V & S_i \in \mathbb{N} & (8)\\
\forall (i,j) \in V^2 & \varphi_{ij} \in \mathbb{N}, \varphi_{ii} = 0 & (9)
\end{cases}$$

The constraints to tackle the routing remain the same as the CP disjunctive formulation with the model $P2$.

A.3. Additional numerical experiments

Extra modeled should be investigated to get around MILP drawbacks, that come from the large number of variables used and from the big-M constraints required for disjunctions of both scheduling and routing.

Cumulative constraint based models have been proved to be efficient in RCPSP resolution (Kreter et al. 2017), and in several RCPSP variants providing results that compete with MILP formulation. Because, the transport operations are fully defined by exchange of resources between activities, the RCPSPR that requires the proper coordination of both activities and routing constraints, presents characteristics that do not favor cumulative model and by consequence resolution (Table A0).

Table A0
CP resolution, disjunctive model vs. cumulative model

		CPLEX CP (disjunctive model P1 and P2)			CPLEX CP (cumulative model P1′ and P2)		
Instances	Optimal solution	BFS	T* (sec.)	TT (sec.)	BFS	T* (sec.)	TT (sec.)
LMQV_U1	74*	74*	122.85	32 885.52	/	t.l.	t.l.

⋆: optimal solution
t.l.: time greater than 48 hours

To obtain a fair comparative study, the two models (disjunctive MILP formulation and CP disjunctive formulation) do not encompass any extra constraint that could strengthen the model and the two models have been solved using the CPLEX 12.7 package running on the same computer. The default parameters are used in the resolution of both MILP and CP: the results are introduced in Table A1 where * is used for optimal solution and $t.l.$ for computational time exceeding 48 hours.

Table A1
GRASP×ELS resolution vs. MILP and CP

	CPLEX MILP (integrated approach)			CPLEX CP (integrated approach)			GRASP×ELS (integrated approach)		
Instances	BFS	T* (sec.)	TT (sec.)	BFS	T* (sec.)	TT (sec.)	BFS	T* (sec.)	TT (sec.)
LMQV_U1	74*	12 348	18 792	74*	122.85	32 885.52	74*	58.6	117.0
LMQV_U2	144*	16 250	17 892	144*	120.18	52 013.33	149	65.9	124.0
LMQV_U3	188*	19 156	23 976	188*	168.85	t.l.	188*	64.8	106.5
LMQV_U4	95*	t.l.	t.l.	95	t.l.	t.l.	100	131.2	208.7
LMQV_U5	192*	66 751	71 172	199	t.l.	t.l.	204	140.4	186.4
LMQV_U6	228*	t.l.	t.l.	228	t.l.	t.l.	231	0.0	237.9
LMQV_U7	97*	11 857	14 580	97*	1 227.69	t.l.	101	15.1	92.0
LMQV_U8	197*	13 019	18 864	197*	56 300.91	t.l.	206	18.1	97.6
LMQV_U9	218*	52 587	56 988	218	4 553.09	t.l.	233	0.1	111.5
LMQV_C1	70*	2 914	4 104	70*	46.00	21 322.73	70*	0.0	0.5
LMQV_C2	118*	2 463	5 364	118*	62.65	16 714.47	118*	0.3	1.0
LMQV_C3	143*	3 796	5 724	143*	37.61	35 038.23	143*	0.0	0.7
LMQV_C4	33*	1 402	1 512	33*	981.33	16 586.54	35	2.7	106.2
LMQV_C5	68*	1 108	1 224	68*	1 405.08	52 238.52	71	2.7	119.0
LMQV_C6	72*	1 021	1 296	72*	219.15	56 232.15	72*	62.8	106.8
LMQV_C7	44*	36 597	54 576	44	t.l.	t.l.	46	0.2	203.4
LMQV_C8	90*	t.l.	t.l.	90	1 787.49	t.l.	96	91.9	140.1
LMQV_C9	94*	t.l.	t.l.	100	t.l.	t.l.	100	0.2	145.9
Avg.		>51 200	>86 400		>52 279	>112 835		36.4	117.0

⋆: optimal solution
t.l.: time greater than 48 hours

The results prove the efficiency of constraint programming formulations first (that permit to obtain numerous optimal solutions) and that the CP optimizer of CPLEX competes with the MILP solver. The coordination between scheduling and routing operations should receive attention to strengthen both models. For example, the difference between starting time of i and starting time of j, for two scheduling operations with a non-null flow ($\varphi_{ij} \geq 0$), could be used to create new transportation constraints by a careful evaluation of the number of loaded and unloaded transport operations required, and by consequence, of the delay between S_j and S_i. The linear constraint $S_j \geq S_i + p_i + M(z_{ij} - 1)$, could be updated with $S_j \geq S_i + p_i + \delta + M(z_{ij} - 1)$ where δ is the extra delay.

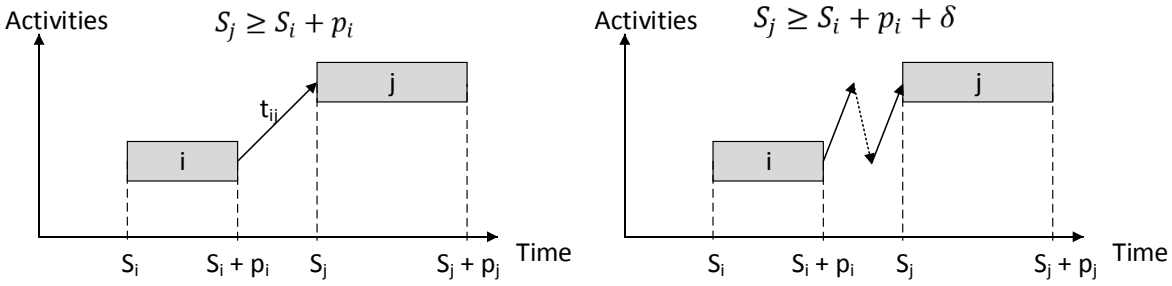


Fig. A1. Illustration of one additional constraint strengthening the model

Table A2 reports the best solutions obtained by CPLEX MILP and CPLEX CP within the time that the GRASP×ELS found its best upper bound. CPLEX MILP failed to provide a solution (except for instance LMQV_U5 and LMQV_C6) whereas CPLEX CP provides 7 solutions, with poor quality regarding the solutions of the GRASP×ELS. For LMQV_U1, CPLEX CP provides a solution of 91 that is strongly larger than 74 which is the best solution found by GRASP×ELS in 58 s. The CPLEX CP provides a first solution in shorten computational time than CPLEX MILP. Such results push us into considering that constraint programming is a promising method that could be used in investigating the RCPSPR.

Table A2
GRASP×ELS resolution vs. MILP and CP within the same time

		GRASP×ELS (integrated approach)		MILP (integrated approach)		CP (integrated approach)
Instances	BKS	BFS	T* (sec.)	(LB)	BFS	BFS
LMQV_U1	74*	74*	58.6	(15)	/	91
LMQV_U2	144*	149	65.9	(30)	/	163
LMQV_U3	188*	188*	64.8	(43)	/	215
LMQV_U4	95*	100	131.2	(16)	/	130
LMQV_U5	192*	204	140.4	(49)	433	230
LMQV_U6	228*	231	0.0	(29)	/	/
LMQV_U7	97*	101	15.1	(15)	/	/
LMQV_U8	197*	206	18.1	(29)	/	/
LMQV_U9	218*	233	0.1	(17)	/	/
LMQV_C1	70*	70*	0.0	(19)	/	/
LMQV_C2	118*	118*	0.3	(27)	/	/
LMQV_C3	143*	143*	0.0	(37)	/	/
LMQV_C4	33*	35	2.7	(12)	/	140
LMQV_C5	68*	71	2.7	(21)	/	/
LMQV_C6	72*	72*	62.8	(32)	110	84
LMQV_C7	44*	46	0.2	(18)	/	/
LMQV_C8	90*	96	91.9	(35)	/	/
LMQV_C9	94*	100	0.2	(35)	/	/
Avg.			36.4			

*: optimal solution
/ : no solution found