A new shortest path algorithm to solve the resource-constrained project scheduling problem with routing from a flow solution

Philippe Lacomme, Aziz Moukrim, Alain A. Quilliot, Marina Vinot

A New Shortest Path Algorithm to Solve the Resource-Constrained Project Scheduling Problem with Routing from a Flow Solution

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
Arc routing

ABSTRACT

In this study, the definition of a RCPSPR (Resource-Constrained Project Scheduling Problem with Routing) solution from a flow solution of the RCPSP is investigated. This new problem consists in defining a solution of RCPSPR that considers both routing and scheduling and that complies with a RCPSP flow, i.e., a solution where the loaded vehicle moves are achieved between activity i and j with a non-null flow. A shortest path algorithm is proposed to solve this problem with a labeling dynamic approach where a label provides all of the information about a solution, including the objective function, the system state and the remaining resources that allow the use of a dominance rule. The system state, described by the label, encompasses both the activities and the vehicle fleet information, including vehicle position and availability dates. Numerical experiments are limited to a comparative study with a proposed linear formulation since no previous publications exist on this problem. A time performance analysis of the proposed algorithm is carried out, proving the efficiency of the algorithm and clearing the way for integration into global iterative optimization schemes that will solve the RCPSPR to optimality.

1. Introduction

Although supply chain decision problems are interrelated, they are often solved sequentially. However, to achieve a highly effective overall system that complies with customers' expectations, coordination among the different stages in the supply chain is necessary. Consequently, the integration of scheduling and routing problems has received increasing attention in the last decade (Moons et al., 2017). Several papers focusing on integrated problems were recently published, including but not limited to Zhang et al. (2016) who deal with the real-world production warehousing case, or Saglam and Banerjee (2017) who focus on batching decisions and different shipping scenarios. In supply chain management, in addition to the integration of production planning and distribution decisions, an effective management of the resources is essential to preserve the competitiveness of companies. This paper focuses on a new integrated problem based on the resource-constrained project scheduling problem with routing constraints.

1.1. Resource-constrained project scheduling problem

The Resource-Constrained Project Scheduling Problem (RCPSP) consists of 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 plus two dummy activities denoted 0 and $n+1$, which define the "project start" and the "project end", respectively. The set of non-dummy activities is identified by $A = \{1, \ldots, n\}$ and some activities are related by precedence constraints. The precedence constraints can be induced by the definition of predecessors in the problem definition (one activity j cannot start before all its predecessors have been achieved) and by definition of constraints due to the resource exchanges. A solution of the RCPSP is fully defined by the activity start times S_i and by a resource supply that complies with the activity requirements. The number of project resources is denoted as q and the set of resource capacities is $R = \{R_1, \ldots, R_q\}$, where $R_k \in \mathbb{N}$. 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.

1.2. RCPSPR definition

The RCPSPR (Resource-Constrained Project Scheduling Problem with Routing) is an extension of the RCPSP where resources are transported from one activity to another using a vehicle. The problem consists of solving both the activity scheduling problem and the vehicle routing problem. This paper focuses on the case of one resource, $q = |R| = 1$, and assumes, without loss of generality, that $b_{ik} = b_i$. The schedule length C_{max} (i. e., the project makespan) is defined by the end of the last transport operation from one activity to the dummy activity $n+1$.

The routing part consists of scheduling trips with a set of vehicles $T = \{1, \ldots, v\}$ sorted in descending order of capacity

[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)

$c_u, u \in \{1, \dots, v\}$. A loaded transportation time t_{ij}^{ux} is defined from activity i to j with a vehicle u loaded with x units of resources. An unloaded transportation time e_{ij}^{u} is also defined from activity i to j with an empty vehicle u. 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 is transferred from activity i to activity j.

The resources transferred from activity i to j are modeled by a transport operation $T_{(i,j,u,x)} = \left(P_{(i,j,u,x)}, D_{(i,j,u,x)}\right)$, which is fully defined by a pickup operation $P_{(i,j,u,x)}$ and a delivery operation $D_{(i,j,u,y)}$. These two operations are defined by:

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

The problem consists in a proper coordination of scheduling and routing operations to minimize the makespan C_{max} (Fig. 1). A routing solution is defined as a set of trips, and each trip is an ordered sequence of loaded transport operations. The earliest starting time of an activity is the latest arrival time of the last vehicle assigned to the transportation of resources required by the activity.

In Fig. 1, an extended Gantt diagram displays a solution with the earliest starting times and the durations of three activities, plus the trips of two vehicles with $c_1 = 3$ and $c_2 = 2$. For example, in Fig. 1, activity 1 has an earliest starting time equal to the arrival time $A_{(0,1,1,3)} = 2$. For activity 3, the earliest starting time is equal to the maximum value between the arrival time of vehicle 2 with two units of resource, $A_{(1,3,2,2)} = 7$ and the earliest completion time of activity 2, $C_2 = 16$, due to a precedence constraint. Let us note that the earliest starting time of activity 2, is the latest arrival time of the vehicles (vehicle 1), meaning that two delivery operation are required to define the starting time of activity 2.

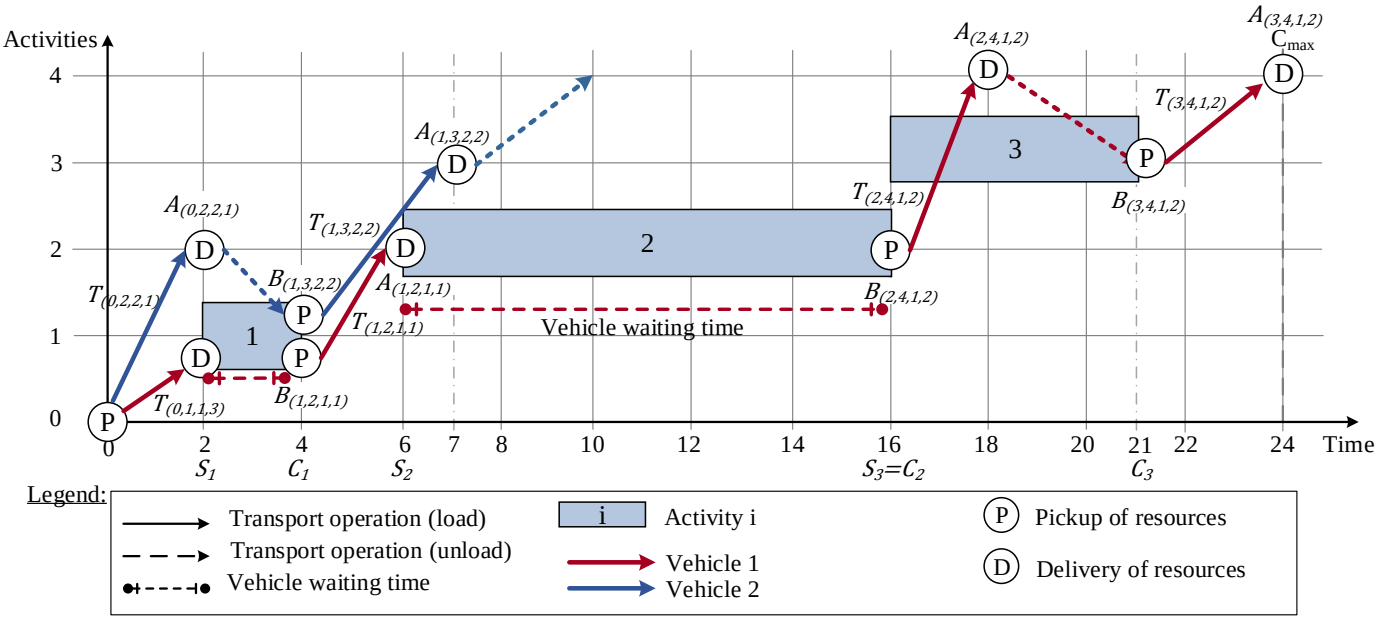


Figure 1. Coordination between routing and scheduling

In this example, vehicle 2 transports one resource from activity 0 to 2 ($T_{(0,2,2,1)}$) and then makes an unloaded move to activity 1 in order to load and transport two resources to activity 3 ($T_{(1,3,2,2)}$). Once the resources are delivered at time 7, vehicle 2 goes back to the depot and arrives at time 10. Figure 2 shows that two deliveries are performed on activity 2, the first delivery operation permits to transport one unit of resource from activity 0 with the vehicle 2 and the second delivery operation permits to transport one unit from activity 1 thanks to vehicle 1. The delivery operation are defined by both a location (activity) and a quantity of resource. The resource delivered at one activity are assumed to be in an input buffer on the activity. Such situation occurs for activity 2, where one unit of resource waits from time 2 to time 6 in the input buffer of activity 2. At time 6, activity 2 can start thanks to the transport operation $T_{(1,2,1,1)}$. This example is used throughout the paper in order to illustrate the proposed algorithm.

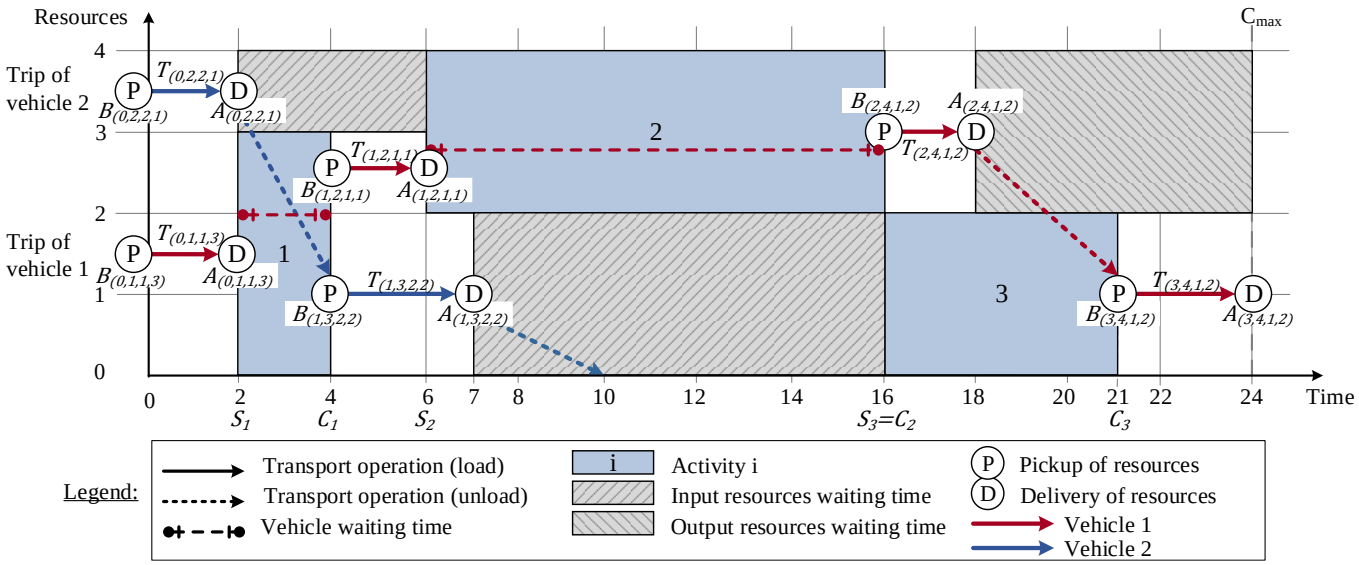


Figure 2. Impact of coordination on the ressource management

1.3. Related problems

Routing constraints are involved in numerous scheduling problems including, for example, the flexible job-shop scheduling problem with transport (Zhang et al., 2012b), the job-shop with transport (Knust, 1999; Lacomme et al., 2013; 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), and the RCPSP with transport (Quilliot et al., 2012). Numerous scheduling approaches take advantage of the disjunctive graph introduced by Roy et al. (1964), which has been extended to tackle transport constraints (see Lacomme et al. (2007) and Zhang et al. (2012a)), where vertex models transport operations and disjunctive arcs are added between operations that require the same resource (vehicle). The coordination between transport and scheduling can be achieved in two possible ways depending on the objective. The first one consists of the explicit modeling of transport from one location to another, and the second one consists in modeling only transport delay (minimal time-lags between machines). Maximal time-lags between activities can model a time window between activities and are used in pickup and delivery resolution approaches for trip evaluation (Cordeau and Laporte, 2003; Firat and Woeginger, 2011).

Transport modeling depends on the vehicle capacity, and a trip is considered to be an ordered sequence of pickup/delivery operations including loaded/unloaded transport operations. Depending on the problem, the transportation time can be job/vehicle load-dependent and can be denoted t_{ij}^{x} for a transport from activity i to j with x resources. Similarly, t_{ij}^{0} (normally denoted $e_{ij} = t_{ij}^{0}$) denotes the duration of an unloaded transport operation. If the transportation times are vehicle-dependent, they can be noted t_{ij}^{ux} where u is the vehicle.

2. Proposition

2.1. RCPSPR modeling

Several formulations were introduced for the RCPSP, including Alvarez-Valdés and Tamarit (1993), Pritsker et al. (1969), Pritsker and Watters (1968) and Dauzère-Pérès and Lasserre, (1995), and more recently, the flow formulation of Artigues et al. (2003) that defines a solution of the RCPSP using an activity-on-node (AON)-flow network defining a so called $G_{AON}^{\varphi}(V, E)$ (Fig. 3). In this graph, there is a vertex in V for each activity. In addition, the set E of resource arcs represents the number of units of the resource directly transferred between two activities. The arc in Fig. 3 between node 0 and i_1 models a resource transfer of φ_{0,i_1} units of the resource between activity 0 and i_1.

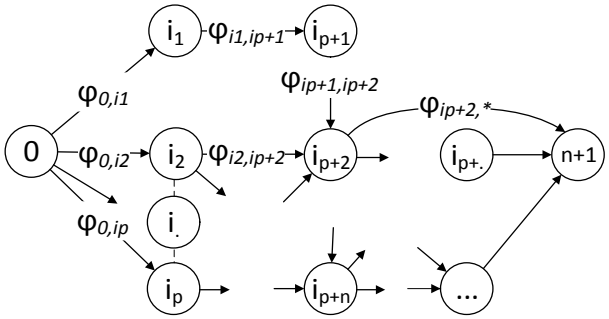


Figure 3. Activity-on-node (AON)-flow network $G_{AON}^{\varphi}(V, E)$

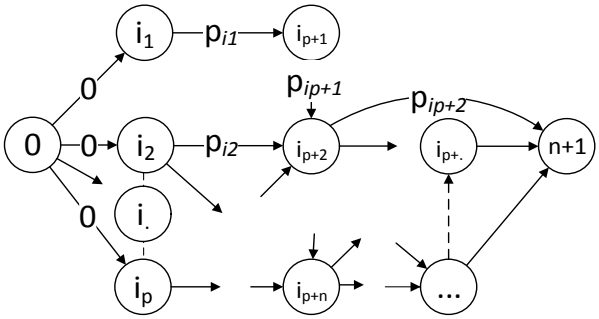


Figure 4. Disjunctive graph $DG_{AON}^{\varphi}(V, E)$

A disjunctive graph $DG_{AON}^{\varphi}(V, E)$ (Fig. 4) is defined considering the (AON)-flow network $G_{AON}^{\varphi}(V, E)$ graph. For each node, $i \in V$, all outgoing arcs $(i, j) \in E$ are weighted by the duration p_i of activity i. If there is an edge $(i, j) \in E$, then $C_i = S_i + p_i \leq S_j$,

since activity j has to be scheduled after activity i. The longest path from 0 to $n+1$ in graph $DG^{\varphi}_{AON}(V,E)$ makes it possible to obtain the earliest starting time of all activities and a critical path. The dashed arc in Fig. 4 models a precedence constraint between the activities.

$G^{\varphi}_{AON}(V,E)$ and $DG^{\varphi}_{AON}(V,E)$ make it possible to define a graph $T^{\varphi}(U,V,E,F)$, which explicitly models the activities and the transport operations of the RCPSPR. In our problem, the arc routing transportation graph $T^{\varphi}(U,V,E,F)$ uses two types of arcs (Lacomme et al., 2005):

- required arc $(i,j) \in E$ corresponding to a positive flow transfer and consequently defining loaded transport operations. A required arc can be serviced by several trips depending on the total demand and on vehicle capacities.
- non-required arcs $(i,j) \in F$ corresponding to a null flow and that can be used for an unloaded transport operation. Non-required arcs make it possible to define deadheading paths from i to j.

In the graph $T^{\varphi}(U,V,E,F)$, the vertices are divided into two disjointed sets:

- U: the pickup node set (in white in Fig 5);
- V: the delivery node set (in gray in Fig 5).

Every arc $e \in E$ connects a vertex in U to a vertex in V to model a loaded transport operation, $|E| = n_{\varphi,}$ and every arc $f \in F$ connects a vertex from V to a vertex U to model an unloaded transport operation (dotted arcs in Fig. 5).

The required arcs (arcs modeling loaded transport operations) are defined by a couple (φ_{ij}, t_{ij}) where φ_{ij} is the number of resources to transport from i to j and where t_{ij} is the transportation time. The valuation of a non-required arc (an arc modeling an unloaded transport operation) is defined by $(0, t_{jv})$.

For convenience, two global dummy nodes are introduced, the first one referred to as # to represent the starting time of the first transport operation, and the second one $*$, the finishing time of the last transport operation.

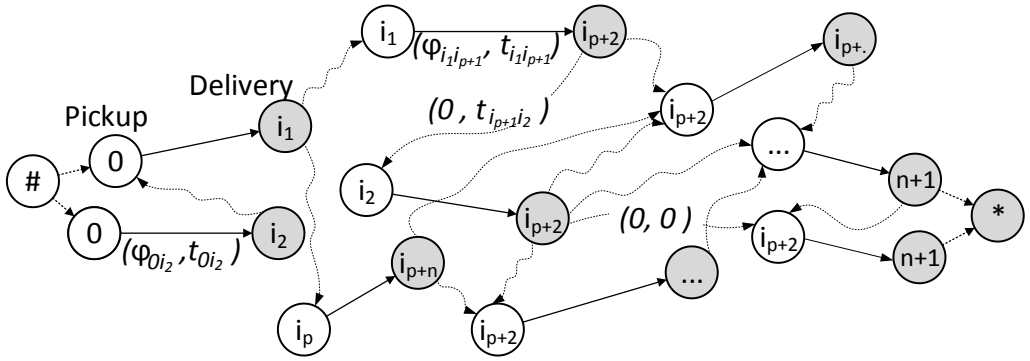


Figure 5. Arc Routing Transportation Graph $T^{\varphi}(U,V,E,F)$

A solution of the RCPSPR is composed of a set of trips assigned to vehicles, knowing that a trip is composed of an ordered set of required arcs and a deadheading path from the destination node of one required arc to the origin node of the next required arc in the trip:

- First, from this point of view, the problem to solve falls into the family of arc routing problems, including, for example, the Chinese Postman Problem (CPP) first introduced by the mathematician Kwan (1962), the Rural Postman Problem (RPP) (Orloff, 1974), and the Capacitated Arc Routing Problem (CARP), that was introduced by Golden and Wong (1981). Contrary to the CARP, where the vehicle load increases during the trip, the pickup/delivery characteristics lead to a specific arc routing problem with no trip infeasibility due to vehicle capacity.
- Second, the problem can be solved using a resolution approach based on a labeling algorithm with an efficient label processing procedure and a specific label definition that makes it possible to obtain a solution of a resource-constrained shortest path problem.

An optimal solution of the RCPSPR defined by the flow can be obtained by execution into $T^{\varphi}(U,V,E,F)$ of a shortest path algorithm with resource constraints between the global dummy nodes # and $*$. The shortest path is defined by an ordered sequence of transport operations with alternation between loaded transport and unloaded transport operations.

2.2. RCPSPR solution defined from a flow

A flow solution defines an acyclic graph $G^{\varphi}_{AON}(V,E)$ where the flow arcs can make it possible to define the transport operations with the quantity of resource. The flow is assumed to comply with the definition of Artigues et al. (2003), and the input/output flow of an activity is assumed to be equal to its required capacity with flow conservation. A flow φ_{ij} between activity i and activity j can exceed the vehicle capacity and can require several transport operations due to the vehicle capacity constraints (c_v).

The new algorithm dedicated to the resource-constrained shortest path problem is defined on a non-ordered set of transport operations and provides an optimal set of trips to build both a routing solution and a scheduling solution that consists of:

- computing the earliest starting times S_i of the activities;
- defining the assignment of vehicles to transport operations;
- ordering the transport operations for each vehicle to define a trip with the departure time and arrival time of the vehicle for each transport operation.

The transport operations are divided into two categories with:

- Loaded transport operation $T_{i,j,*,\varphi_{ij}}$ from activity i to j, modeled by an arc from i to j where the valuation is

composed of the flow φ_{ij} (that must be transported from activity i and activity j) and the transportation time t_{ij}.

- Unloaded transport operation, modeled by an arc (dotted arcs in Fig. 5) valuated with 0 for the flow and t_{ij} for the transportation time.

2.3. A New Shortest Path Algorithm and its application to arc routing

The resource-constrained shortest path problem considered in this paper makes it possible to define an optimal solution of the RCPSPR in the graph $T^{\varphi}(U,V,E,F)$. Each solution (Fig. 6) is composed of a set of trips (modeled by a path in $T^{\varphi}(U,V,E,F)$ from node # to node $*$) passing through every arc with a non-null flow. The solution is optimal if it minimizes the arrival time of all the vehicles on the node $*$ with respect to all the constraints. Figure 6 shows two trips assigned to two vehicles. The trip of vehicle 1 is composed of six transport operations, whereas the trip of vehicle 2 consists of three transport operations. These two trips are interrelated due to some activities in both trips, e.g. activity i_{p+2}. The two delivery nodes i_{p+2} must be scheduled before the two pickup nodes $i_{p+2,}$ and the departure times of the two loaded transport operations from the two pickup nodes i_{p+2} are greater or equal to the completion time of the activity i_{p+2}.

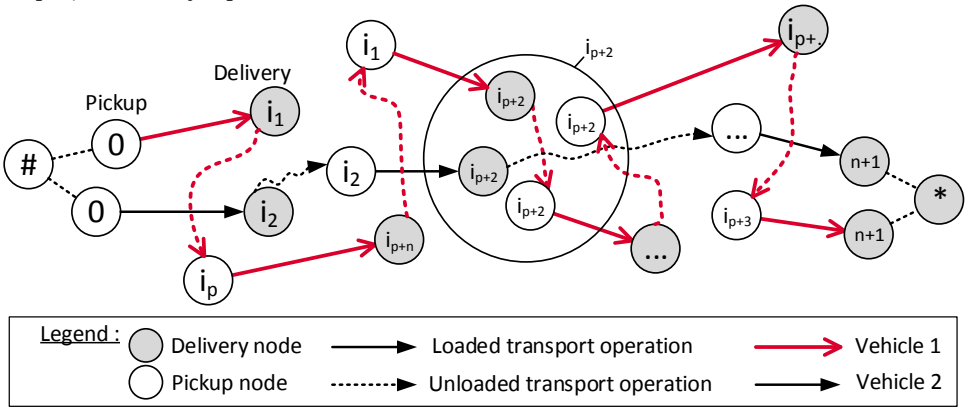


Figure 6. A solution with two trips in the graph $T^{\varphi}(U,V,E,F)$

Each partial solution (also denoted for convenience as a solution) is modeled by a path represented by a label $L(f,S,R)$ i.e., a data structure that provides all of the information about the solution such as the objective function, the system state and the remaining resources that allow the use of a dominance rule. A partial solution is defined by a label where all the flows between operations have not yet been transported by vehicles, meaning that all activities are not yet scheduled. Conversely, a final solution is a label where all flows have been transported, i.e., a label where all activities have been scheduled.

For each node i of the graph $T^{\varphi}(U,V,E,F)$, Y_i is the ordered set of unprocessed labels, i.e., paths that have not been extended along all arcs (i,j) leading to feasible paths. The set P_i contains labels that are required to be kept, i.e., P_i defines a set of non-dominated labels on node i. The labels are sorted into non-decreasing order of the total loaded transportation time t_{ij} from activity i to j not yet serviced according to the required flow, and Z_i is the restriction of Y_i to the N_L first labels.

For each node i, the distance $D[i]$ is the longest path in terms of the number of arcs from i to the dummy node * . In other words, $D[i]$ is a distance between a partial solution and a solution of the problem and is representative of the computational effort required to obtain a final solution. The distances are preprocessed and are directly used in the following algorithm.

The use of a dominance rule is optional in the sense that the algorithm otherwise enumerates all feasible paths starting at node #, but dominance is crucial in the design of one efficient resource-constrained shortest path algorithm to identify paths that do not need to be extended.

The outline of the resource-constrained shortest path algorithm for solving the RCPSPR is presented in Algorithm 1. The problem consists of computing a minimum-cost feasible path ending at the global dummy nodes * where a set of non-dominated labels (each label modeling a partial solution) is stored. The algorithm starts with one initialization step (lines 12-15) where a label L_0 is stored on the global dummy node # (that models the depot node of vehicles). Lines 16-34 define the outer loop where labels are propagated along feasible-path constraints. The algorithm terminates when no further unprocessed label on the node exists, i.e., when all labels giving the non-dominated paths from vertex # to * have been created, which is defined by an empty queue Λ. The algorithm selects a final solution Sol (line 35) that minimizes the objective function f.

```
      Procedure name: New Shortest Path Algorithm
1.  procedure Shortest_Path
2.  input parameters
2.    T^φ(U,V,E,F): Arc routing transportation graph
3.    UB: Upper Bound of the problem
4.    LB_i: Lower Bound to reach the final node from node i
5.  output parameters
6.    Sol: Solution
7.  global parameter
8.    Λ: Queue
9.    n_vehicle: number of vehicles
10.   N_L: maximum number of unprocessed labels into Z_{i ∈ U∪V}
11.  begin
12.    Definition of L_o
13.    for node_i ∈ U ∪ V do
14.      if (λ(node_i) = 0) then Z_node_i = {L_o}, PUSH(Λ,node_i) else P_node_i = {∅} endif
15.    endfor
16.    while (Λ != ∅) do
17.     node_i := POP(Λ)
18.     for L_i ∈ Z_node_i do                          //for each label unprocessed on node i
19.       for node_j ∈ succ(node_i)do
20.         if PATH(L_i,node_j) is feasible then       //resources and precedence constraints
21.           for v:= 1 to n_vehicle do
22.              call CREATE_LABEL(L_w,v, L_i,node_i,node_j)
23.              if (DOMINATE(L_w,node_j)) and (CHECK_UB_LB(L_w,LB_node_j,UB)) then
24.                 call INSERT_LABEL(L_w,Z_node_j), call PUSH(Λ,node_j),
25.                 call APPLY_DOMINANCE(L_w,Z_node_j)
26.              endif
27.              if (λ(node_j) = *) then call CHECK_SOL(L_w,UB) endif
28.           endfor
29.         endfif
30.       endfor
31.     Z_node_i = Z_node_i \ L_i
32.     P_node_i = P_node_i ∪ L_i
33.     endfor
34.   endwhile
35.   Sol := BEST_SOL (T^φ) //save the best solution
36. end
```

Algorithm 1. New shortest path procedure.

The Boolean function $DOMINATE(i,j)$ (line 23) return true if the label i is not dominated by an unprocessed label on node j.

$$DOMINATE(i,j) = \nexists k \in Z_j, k \ll i \qquad (1)$$

The algorithm also uses the function $CHECK\_UB\_LB(L,LB,UB)$ (line 23), which returns true if the label L cannot be pruned, and false if the objective function of the label L plus the lower bound LB does not fit the upper bound UB.

The label feasibility is checked considering the remaining resources in the function $PATH(L_i,node\_j)$. The propagation rule to create a new label (detailed below) is achieved by the procedure $CREATE\_LABEL(L_w,v,\ L_i,node_i,node_j)$ (line 22), which makes it possible to obtain a new label L_w from label L_i considering vehicle v. The procedure $INSERT\_LABEL(L_w,Z_{node\_j})$ (line 24) carries out the insertion of the label L_w in the ordered set of unprocessed labels Z_{node_j} that take the maximum number of unprocessed labels authorized on each node N_L into account. Then, with the function $APPLY\_DOMINANCE(L_w,Z_{node\_j})$ (line 25), all labels on Z_{node_j} that are dominated by L_w are removed from Z_{node_j}.

The $CHECK\_SOL(L_w,UB)$ (line 27) function updates the upper bound of the problem for each new solution (defining a path through every arc with a non-null flow) on the final node. Finally, the best solution at the end of the algorithm is given by the function $BEST\_SOL\ (T^{\varphi})$ (line 35) from the best final label in T^{φ}.

The resource-constrained shortest path algorithm must encompass the management of both the availability of vehicles and that of activities, and includes the following features:

- a label definition to encompass the system state;
- creation of the initial label;
- a propagation rule to create a new label;
- a label feasibility check;
- a dominance rule to save only non-dominated labels on the node.
- a label propagation selection rule required to choose the next path (next label) to be extended.

Let us define a function that makes it possible to assign a number in $[1..n_{\varphi}]$ to each arc.

$$\beta: e \in E \rightarrow [1..n_{\varphi}],$$
$$\beta(e \in E) = i$$
$$\forall e_1 \in E, \forall e_2 \in E,\ e_1 \neq e_2, \beta(e_1) \neq \beta(e_2).$$

Another notation is also used to identify the pickup node and the delivery node of each numbered arc $i \in [1..n_{\varphi}]$. Let us denote i^+ as the position number of the pickup node in graph G and i^- as the position of the delivery node of arc i. Let us define λ as the

activity number where a pickup operation (resp. delivery operation) is achieved from position i^+ (resp. i^-) such that $\lambda(i^-)$ or $\lambda(i^+)$ gives the activity where the operations are achieved.

Label definition

A label $L = (f, S, R)$ represents a partial or final solution and L_i^j denotes the i^{th} label on node $j = j^-$ or $j = j^+$, which is composed of three parts:

- The objective function value of the solution, i. e., the maximal completion time of the vehicles.
- The system state S that encompasses the vehicle fleet and the activities. The system state can be described by:
  - a $3v$-uplet V_i^j defining the vehicle departure and arrival times, and positions, $V_i^j = \left(P_{ij}^1, \dots, P_{ij}^v, A_{ij}^1, \dots, A_{ij}^v, B_{ij}^1, \dots, B_{ij}^v \right)$, 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), $A_{ij}^1, \dots, A_{ij}^v$ are the arrival times, and $B_{ij}^1, \dots, B_{ij}^v$ are the departure times of the vehicles;
  - a $n+1$-uplet S_i^j defining the earliest starting time of each activity $S_i^j = (S_{ij}^0, \dots, S_{ij}^{n+1})$.
- The resource state R remaining on arcs modeling loaded transport operations (required arcs) and the remaining number of resources to start each activity:
  - a n_φ-uplet $\varphi_i^{R,j} = (\varphi_{ij}^{R,1}, \dots, \varphi_{ij}^{R,n_\varphi})$ defining the remaining resources of the arc $e \in E$; $\varphi_{ij}^{R,\beta(e)}$ not yet transported;
  - a $n+1$-uplet $b_i^{R,j} = (b_{ij}^{R,0}, \dots, b_{ij}^{R,n+1})$ defining the remaining resources $b_{ij}^{R,k}$ for activity k not yet transported.

Creation of the initial label

An initial label L is stored on node # and propagated to all nodes $j = j^+$, where $\lambda(j^+) = 0$. To comply with the following definition, all the values are initialized:

$$\forall u \in T,\ P_{1j}^u = 0$$
$$\forall u \in T,\ A_{1j}^u = 0$$
$$\forall u \in T,\ B_{1j}^u = 0$$
$$\forall k \in V^*,\ S_{1j}^k = -\infty \text{ and } S_{1j}^0 = 0$$
$$\forall i \in [1..n_\varphi],\ \varphi_{1j}^{R,i} = \varphi_{\lambda(j^+),\lambda(j^-)}$$
$$\forall k \in V,\ b_{qj}^{R,k} = b_{\mathrm{k}}$$
$$f_{1j} = 0$$

Propagation rule for a label on a pickup node (arc modeling a loaded transport operation)

A propagation function defined by $f: \mathrm{L} \otimes \mathrm{T} \rightarrow \mathrm{R}$ that makes the propagation of a new label $L_q^{i^-}$ from one label $L_p^{i^+}$ and a loaded transport operation defined by $(\varphi_{mk}, t_{mk})\ /m = \lambda(i^+)$ and $k = \lambda(i^-)$ possible.

The new label L_q^j for node number $j = i^-$ is defined from the label L_p^i $i = i^+$ using the vehicle u with the following updates:

$$P_{qj}^u = \lambda(j)$$
$$A_{qj}^u = B_{pi}^u + t_{\lambda(i),\lambda(j)}$$
$$B_{qj}^u = A_{qj}^u$$
$$S_{qj}^{\lambda(j)} = max(A_{qj}^u;\ S_{pi}^{\lambda(j)})$$
$$\varphi_{qj}^{R,i} = \varphi_{pi}^{R,i} - min(\varphi_{pi}^{R,i};\ c_u)$$
$$b_{qj}^{R,\lambda(j)} = b_{pi}^{R,\lambda(j)} - min(\varphi_{pi}^{R,i};\ c_u)$$
$$f_{qj} = max(f_{pi};\ A_{qj}^u)$$

If $b_{qj}^{R,\lambda(j)} = 0$, the propagation rule also updates all the starting times of the successors of activity $\lambda(i)$, i.e., $\forall s \in succ(\lambda(i)), S_{qj}^s = max(S_{pi}^s;\ S_{qj}^{\lambda(j)} + p_{\lambda(j)})$. This update ensures that precedence constraints hold. For the label L_p^i, all assignments to vehicles are investigated, leading to v labels. These labels are included or not depending on the set of non-dominated labels previously stored on the node j in $T^\varphi(U, V, E, F)$.

Propagation rule for a label on a delivery node (arc modeling an unloaded transport operation)

A propagation function defined by $f: \mathrm{L} \otimes \mathrm{T} \rightarrow \mathrm{R}$ that makes the generation of a new label $L_q^{j^-}$ from one label $L_p^{i^+}$ and an unloaded transport operation defined by $(0, t_{mk})/m = \lambda(j^-)$ and $k = \lambda(i^+)$ possible.

The new label L_q^j with $j = i^+$ is defined from the label L_p^i, where $i = j^-$ using vehicle u with the following updates:

$$P_{qj}^u = \lambda(j)$$
$$A_{qj}^u = B_{pi}^u + e_{\lambda(i),\lambda(j)}$$
$$B_{qj}^u = max(A_{qj}^u;\ S_{pi}^{\lambda(j)} + p_{\lambda(j)})$$
$$f_{qj} = max(f_{pi};\ A_{qj}^u)$$

Label feasibility check

The first and the second conditions detailed below only hold for label $L_q^{i^-}$ stored at delivery node $i = i^-$ propagated to a pickup node $j = j^+$. Both conditions are linked to the resource state. The **first condition** takes advantage of the flow. If the flow $\varphi_{qi}^{R,j} = 0$, i.e., all resources between $\lambda(j^+)$ and $\lambda(j^-)$ were transported, then an unloaded transport operation to activity $\lambda(j^+)$ must be forbidden. The **second condition** holds since an unloaded transport operation can only be achieved to an activity $\lambda(j^+)$ previously scheduled, i.e., for which all the resources were delivered $b_{qi}^{R,\lambda(j)} = 0$. If $b_{qi}^{R,\lambda(j)} \neq 0$, no unloaded transportation operation to $\lambda(j^+)$ must be scheduled.

Dominance rule

Whenever one or several new labels are created (by the propagation rule), they are compared for dominance with the non-dominated labels that are stored at the destination node. Considering two labels L_p^i and L_q^i, L_p^i is defined as dominant as regards L_q^i ($L_q^i \ll L_p^i$) if the following conditions hold:

Condition 1. All vehicles have the same location:

$$\forall u \in T, \qquad P_{ip}^u = P_{iq}^u$$

Condition 2. $\forall u \in T,\ B_{ip}^{u} \leq B_{iq}^{u}$
Condition 3. $\forall u \in T,\ A_{ip}^{u} \leq A_{iq}^{u}$
Condition 4. $\forall k \in [1..n_{\varphi}], \varphi_{ip}^{R,k} \leq \varphi_{iq}^{R,k}$
Condition 5. $S_{ip}^{n+1} \leq S_{ip}^{n+1}$
Condition 6. $\exists u \in T,\ B_{ip}^{u} < B_{iq}^{u}$
or $\exists u \in T,\ A_{ip}^{u} < A_{iq}^{u}$
or $\exists k \in [1..n_{\varphi}], \varphi_{ip}^{R,k} < \varphi_{iq}^{R,k}$
or $S_{ip}^{n+1} < S_{ip}^{n+1}$

If L_p^i is not dominant as regards L_q^i ($L_q^i \lll L_p^i$), this does not imply that L_q^i is dominant as regards L_p^i. If $L_q^i \lll L_p^i$ and $L_p^i \lll L_q^i$, 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 subset of labels while maintaining algorithm optimality. An additional time-saving approach consists in limiting the maximal number N_L of labels stored on each node. Such restrictions, in addition to the dominance rule, can strongly reduce the CPU time but can yield to a sub-optimal final solution.

Label propagation selection rule

The algorithm relies on a queue referred to as Λ that supports the following operations:

- Push (Λ, i): adds the node number i to the queue and guaranties that each node number i cannot be added to the queue more than once;
- Pop (Λ): removes the node with the minimal $D[i]$ and the largest number of labels.

By adopting the Λ order for the node selection (line 17), the algorithm favors propagation of labels that are more prone to create a solution within an efficient time delay.

2.4. Example

In this section an example is introduced in order to build a graph $T^{\varphi} = (U, V, E, F)$ to illustrate some steps of the algorithm. The example introduced below is composed of three activities and two dummy activities modeling the depot where four resources are available. The duration of the activities is given in Table 1 and the distance matrix is introduced in Table 2. For the routing part, two vehicles are available, vehicle 1 with a capacity of 3 and vehicle 2 with a capacity of 2. Both vehicles are assumed to have a traveling speed of 1.

Table 1
Information about the activities.

Activity	Duration	Resource requirement	Successors
0	0	/	1,2,3,4
1	2	3	4
2	10	2	3,4
3	5	2	4
4	0	/	/

Table 2
Matrix distance between the activities.

	0	1	2	3	4
0	0	2	2	3	0
1	2	0	2	3	2
2	2	2	0	5	2
3	3	3	5	0	3
4	0	2	2	3	0

Let us assume that a specific algorithm solves the flow problem defining a flow-network solution $G_{AON}^{\varphi}(V, E)$ (Fig. 7) and let us define the graph $T^{\varphi}(U, V, E, F)$ on this solution, where $U = \{0,0,1,1,2,3\}$, $V = \{1,2,2,3,4,4\}$, $|E| = 6$, and $|F| = 20$, introduced in Fig. 8.

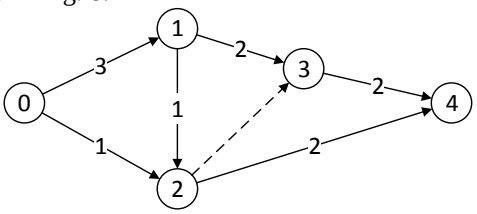
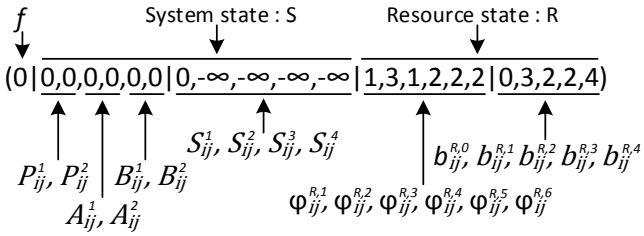


Figure 7. Example of an activity-on-node (AON)-flow network $G_{AON}^{\varphi}(V, E)$.

For the **first step** (Fig. 8), the graph is initialized with a label $L^{\#}$ leading to two labels stored on node 1^+ and 2^+, corresponding to activity 0 as $\lambda(1^+) = 0$ and $\lambda(2^+) = 0$. The two labels $L_1^{1^+}$ and $L_1^{2^+}$ are both equal to:



with respect to the label definition, $i = 1$ and $j = \{1^+, 2^+\}$. The queue is equal to $\Lambda = [1^+, 2^+]$.

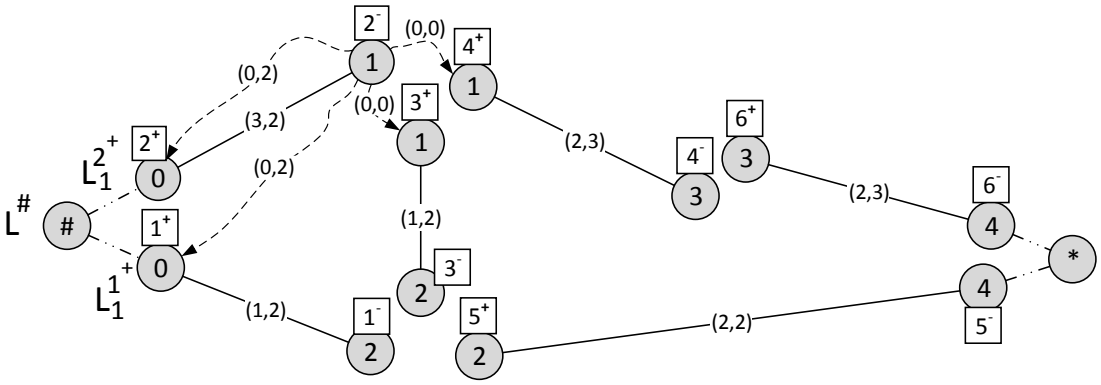


Figure 8. Initialization of the graph $T^{\varphi}(U, V, E, F)$.

In the **second step**, the node number 2^+ is removed from the queue and all the labels on node 2^+ are propagated on node 2^-. In this step, two labels are created (Fig. 9):

- $L_1^{2^-} = (2|1,0,2,0,2,0|0,2,-\infty,-\infty,4|1,0,1,2,2,2|0,0,2,2,4)$
- $L_2^{2^-} = (2|0,1,0,2,0,2|0,2,-\infty,-\infty,4|1,1,1,2,2,2|0,1,2,2,4)$

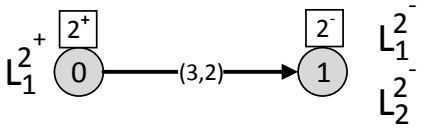


Figure 9. Step 2: Propagation of the label on arc number 2 in $T^{\varphi}(U, V, E, F)$.

The creation of label $L_1^{2^-}$ can be designed to take label $L_1^{2^+}$, the arc $(0,1)$ with the information $(3,2)$ and the assignment of vehicle 1 into account.

Label $L_1^{2^-}$ is created considering the propagation rule (loaded transport operation):

$P_{1,2^-}^1 = \lambda(2^-) = 1$

$A_{1,2^-}^1 = B_{1,2^+}^1 + t_{0,1} = 0 + 2 = 2$

$B_{1,2^-}^1 = A_{1,2^-}^1 = 2$

$S_{1,2^-}^1 = max\left(S_{1,2^+}^1;\ A_{1,2^-}^1\right) = max(-\infty; 2) = 2$

$\varphi_{1,2^-}^{R,2} = \varphi_{1,2^+}^{R,2} - min\left(\varphi_{1,2^+}^{R,2};\ c_1\right) = 3 - min(2; 3) = 0$

$b_{1,2^-}^{R,1} = b_{1,2^+}^{R,1} - min\left(\varphi_{1,2^+}^{R,2};\ c_1\right) = 3 - min(2; 3) = 0$

and because $b_{1,2^-}^{R,1} = 0$, the earliest starting time of all successors of activity 1 should be updated:

$S_{1,2^-}^* = S_{1,2^-}^1 + p_1 = 2 + 2 = 4$

The two labels created are not comparable due to the position of the vehicles. They are therefore both stored on node 2^-. The queue is updated and is equal to $\Lambda = [1^+, 2^-]$.

In the **third step**, the node number 2^- is removed from the queue and the labels on node 2^- are propagated on nodes $1^+, 2^+, 3^+, 4^+$ with a feasibility check. This label propagation concerns unloaded transport operations in the dashed arcs in Fig. 8.

Ten labels are created:

- Four on node 1^+:
  $L_1^{1^+} = (4|0,0,4,0,4,0|0,2,-\infty,-\infty,4|1,0,1,2,2,2|0,0,2,2,4)$
  $L_2^{1^+} = (2|1,0,2,0,2,0|0,2,-\infty,-\infty,4|1,0,1,2,2,2|0,0,2,2,4)$
  $L_3^{1^+} = (2|0,1,0,2,0,2|0,2,-\infty,-\infty,4|1,1,1,2,2,2|0,1,2,2,4)$
  $L_4^{1^+} = (4|0,0,0,4,0,4|0,2,-\infty,-\infty,4|1,1,1,2,2,2|0,1,2,2,4)$
- Two on node 2^+:
  $L_1^{2^+} = (2|0,1,0,2,0,2|0,2,-\infty,-\infty,-\infty|1,1,1,2,2,2|0,1,2,2,4)$
  $L_2^{2^+} = (4|0,0,0,4,0,4|0,2,-\infty,-\infty,-\infty|1,1,1,2,2,2|0,1,2,2,4)$
- Two on node 3^-:
  $L_1^{3^+} = (4|1,0,2,0,4,0|0,2,-\infty,-\infty,4|1,0,1,2,2,2|0,0,2,2,4)$
  $L_2^{3^+} = (4|1,1,2,2,2,4|0,2,-\infty,-\infty,4|1,0,1,2,2,2|0,0,2,2,4)$
- Two on node 4^-:
  $L_1^{4^+} = L_1^{3^+}$ and $L_2^{4^+} = L_2^{3^+}$

Table 3
Details of all the labels of the optimal solution in Fig. 10.

Node	List of labels	
#	$L_{\cdot}^{\#} =$	(0|0,0,0,0,0,0|0, −∞, −∞, −∞, −∞|1,3,1,2,2,2|0,3,2,2,4)
1^+	$L_{\cdot}^{1^+} =$	(2|1,0,2,0,2,0|0,2, −∞, −∞,4|1,0,1,2,2,2|0,0,2,2,4)
1^-	$L_{\cdot}^{1^-} =$	(2|1,2,2,2,2,2|0,2,2, −∞,4|0,0,1,2,2,2|0,0,1,2,4)
2^+	$L_{\cdot}^{2^+} =$	(0|0,0,0,0,0,0|0, −∞, −∞, −∞, −∞|1,3,1,2,2,2|0,3,2,2,4)
2^-	$L_{\cdot}^{2^-} =$	(2|1,0,2,0,2,0|0,2, −∞, −∞,4|1,0,1,2,2,2|0,0,2,2,4)
3^+	$L_{\cdot}^{3^+} =$	(4|1,3,2,7,4,7|0,2,2,7,11|0,0,1,0,2,2|0,0,1,0,4)
3^-	$L_{\cdot}^{3^-} =$	(6|2,3,6,7,6,7|0,2,6,16,16|0,0,0,0,2,2|0,0,0,0,4)
4^+	$L_{\cdot}^{4^+} =$	(4|1,1,2,4,2,4|0,2,2, −∞,4|0,0,1,2,2,2|0,0,1,2,4)
4^-	$L_{\cdot}^{4^-} =$	(6|1,3,2,7,2,7|0,2,2,7,11|0,0,1,0,2,2|0,0,1,0,4)
5^+	$L_{\cdot}^{5^+} =$	(16|2,3,16,7,16,7|0,2,6,16,16|0,0,0,0,2,2|0,0,0,0,4)
5^-	$L_{\cdot}^{5^-} =$	(18| ∗ ,3,18,7,18,7|0,2,6,16,18|0,0,0,0,0,2|0,0,0,0,2)
6^+	$L_{\cdot}^{6^+} =$	(21|3,3,20,7,21,7|0,2,6,16,18|0,0,0,0,0,2|0,0,0,0,2)
6^-	$L_{\cdot}^{6^-} =$	(24| ∗ ,3,24,7,24,7|0,2,6,16,24|0,0,0,0,0,0|0,0,0,0,0)
*	$L_{\cdot}^{*} =$	(24| ∗,∗ ,24,10,24,10|0,2,6,16,24|0,0,0,0,0,0|0,0,0,0,0)

At the end of the algorithm, i.e., when all labels have been propagated (the queue Λ is then empty), a set of non-dominated solutions are obtained. The final solutions that minimize the makespan are optimal since they minimize the objective function (first parameter in the labels). An optimal solution (solution minimizing the makespan) is introduced in Fig. 10 considering the trip of vehicle 1 in Fig. 10.a and the trip of vehicle 2 in Fig. 10.b. All the labels are fully described in Table 3. In Fig. 1 and 2, an extended Gantt diagram displays this optimal solution.

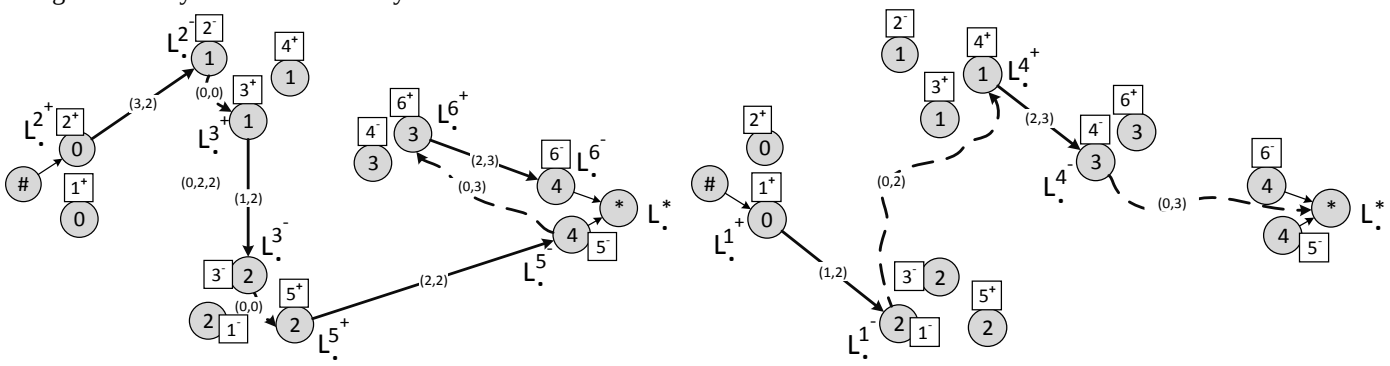
2.5. Discussion

The algorithm proposed to solve the RCPSPR from a flow solution takes advantage of a resource-constrained shortest path algorithm.

This algorithm is dedicated to the resolution of an arc routing problem with specific features based on the fact that the

scheduling and the routing are interrelated with shared resource management. In the problem of interest here, the transport operations are first linked by precedence constraints due to the flow definition and, second, model the required arcs with a quantity that can eventually exceed the vehicle capacity. To the best of our knowledge, there is no specific method in the arc routing community that simultaneously tackles these constraints and that could provide time-saving implementation.

This algorithm first attempts to optimally transform a RCPSP flow solution into a RCPSPR solution and is the first step towards defining an efficient framework based on the indirect modeling of solutions using the flow. This is a theoretical contribution with a practical application.



a. Trip of vehicle 1 b. Trip of vehicle 2

Figure 10. Representation of the two trips of an optimal solution on the graph $T^{\varphi}(U,V,E,F)$.

3. Numerical experiments

The aim of these experiments is to provide feedback on the efficiency of the algorithm to compute an optimal RCPSPR solution from one RCPSP solution. The efficiency is analyzed considering the CPU time required in practice and the total number of labels generated during the process.

All experiments were carried out on a single thread C program, using Visual Studio and a Windows 7 operating system on a Dell Optiflex9020 with an Intel Core i7-4770 CPU 3.4 GHz and 16 Gb of RAM, meaning approximately 2671 Mflops (see Dongarra et al., 2014). The CPLEX experiments were carried out on the same computer. In order to ensure fair future comparative studies, all results and one example are available at the following web page:

http://fc.isima.fr/~vinot/Research/RCPSPR_Flow.html

3.1. New set of instances

To the best of our knowledge, no instance dealing with this problem is available. Consequently, numerical experiments are based on a new set of small-scale instances composed of 18 instances with six activities (plus the dummy nodes). In these instances, the location of the activities can be divided into two configurations: the first one with a uniform distribution of the activities and the second one with two clusters.

Tables 4 and 5 introduce the number of activities to be scheduled, but this number does not include information on the number of transport operations. The number of transport operations depends on the flow modeling a RCPSP solution and on the vehicle capacities. Column $LB(n)$ is the lower bound of the total number of operations to be scheduled, considering both n and the minimal number of arcs with a non-null flow, i.e., $n+1$, obtained with an ordered sequence of activities.

Table 4
Small-scale instance characteristics.

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

The resource requirement and availability, the capacity of the vehicles and the ratio are given in Table 4. The ratio is defined as the ratio between the average duration of an activity and the average duration of a transport operation per vehicle, which is representative of the relative importance of the scheduling vs. the routing. A ratio greater than one means that the scheduling processing time represents a greater amount of time than the routing, and a ratio lower than one implies the reverse.

A new set of medium-scale instances composed of nine instances with 30 activities (plus the dummy nodes) is also introduced. In these instances, the location of the activities is divided into three configurations: the first one with a uniform distribution of the activities and the depot located at the center of the location, the second one with two clusters and the depot located at the center of the location, and the third one with two clusters and the depot location in a cluster (Table 5).

Due to the combinatorial nature of the RCPSP, obtaining optimal solutions using exact methods for problems with over 60 or so activities becomes intractable and, hence, impractical (Valls et al., 2005). Moreover, the RCPSPR is an extension of the RCPSP with the management of a fleet of vehicles where the decision variables encompass both RPCSP variables and a set of variables for the routing problem. In this new set of medium instances, more than 60 operations have to be scheduled ($LB(n) = 61$). The optimal resolution of this set of instances is challenging due, first, to the large number of operations to be scheduled, including both the activities and the transport operations and, second, to the coordination between the activities and the transport operations, including the assignment of vehicles to the transport operations.

All the details of the instances are available at the web page.

Table 5
Medium-scale instance characteristics.

Instances	n	LB(n)	Location	Resource requirement	Resource availability	Vehicle capacity	Ratio
LMQV_J30_U1	30	61	uniform	[1;10]	13	{13;13}	0.7
LMQV_J30_U2	30	61	uniform	[1;10]	14	{8;6}	0.3
LMQV_J30_U3	30	61	uniform	[1;10]	13	{13;9}	1.0
LMQV_J30_C1	30	61	clusters	[1;10]	15	{15;15}	0.8
LMQV_J30_C2	30	61	clusters	[1;10]	11	{7;5}	0.3
LMQV_J30_C3	30	61	clusters	[1;10]	12	{12;9}	0.9
LMQV_J30_CC1	30	61	clusters	[1;10]	13	{13;13}	0.6
LMQV_J30_CC2	30	61	clusters	[1;10]	14	{8;6}	0.3
LMQV_J30_CC3	30	61	clusters	[1;10]	13	{13;8}	1.1

3.2. Performance of the shortest path algorithm

The effectiveness of the shortest path algorithm can be evaluated on small-scale instances by considering a restriction of Y_i to 500 labels per node, with $r = 10$ replications per instance. One replication models one randomly generated flow (with one replication considering the flow leading to the optimal RCPSP solution). The shortest path algorithm is compared to an optimal resolution with IBM ILOG CPLEX 12.6 and a linear formulation of the problem (Lacomme et al., 2017).

For each instance, $\overline{h_C(x,\mathrm{r})}$ defines the average solution cost with CPLEX, and $\overline{t_C(x,r)}$ the average computational time required to find the optimal solution with CPLEX. Similarly, $\overline{h_{SP}(x,\mathrm{r})}$ denotes the average best-found solution cost using the shortest path algorithm (restriction of Y_i can lead to a suboptimal solution), and $\overline{t_{SP}(x,r)}$ the average resource-constrained shortest path algorithm time to the best-found solution. The notation $\overline{g(x,n)}$ denotes the gap between $\overline{h_{SP}(x,r)}$ and $\overline{h_C(x,r)}$ for the instance x, and $\overline{g(x,r)}$ is an estimation of the average gap considering n replications. The number of optimal solutions

found by the shortest path algorithm for the instance x with n replications is denoted $n_{SP}(x,n)$. Let us denote $\overline{g(.,r)}$ as the estimation of the average gap considering the 18 instances, $\overline{t_*(.,r)}$ (resp. $\overline{t_{SP}(.,r)}$, and $\overline{t_C(.,r)}$) the estimation of the average computational time (resp. of CPLEX and of the shortest path), $\overline{n_{SP}(.,r)}$ the average number of optimal solutions found over the 18 instances, and $n_{SP}(.,.) = \sum_{x=1}^{x=18} n_{SP}(x,r)$, i.e., the total number of optimal solutions found during the 10 replications and the 18 instances.

The average resource-constrained shortest path computational time $\overline{t_{SP}(.,r)}$ in Table 6 is approximately 8.2 seconds, approximately two times lower that the CPLEX computational time $\overline{t_C(.,r)}$, which is approximately 17.8 seconds (Table 6). The average number of optimal solutions found by the resource-constrained shortest path algorithm is greater than 9.9, which means that the percent of optimal solutions found is close to 100%. It can be observed that the resource-constrained shortest path algorithm finds the optimal solution for 176 flows out of 180 ($n_{SP}(.,.)$), with an average gap of 0.04%.

Table 6
Average shortest path efficiency with a restriction of Y_i to 500 labels per node ($r = 10$ flows per instance) for small-scale instances.

	CPLEX optimal resolution		Shortest path optimal resolution			
Instances	$\overline{h_C(x,r)}$	$\overline{t_C(x,r)}$	$\overline{h_{SP}(x,r)}$	$n_{SP}(x,r)$	$\overline{t_{SP}(x,r)}$	$\overline{g(x,r)}$
LMQV U1	101.5	46.6	101.5	10/10	10.6	0.00 %
LMQV U2	198.5	37.2	198.7	9/10	10.2	0.09 %
LMQV U3	246.6	29.0	246.6	10/10	11.9	0.00 %
LMQV_U4	123.3	31.0	123.3	10/10	3.6	0.00 %
LMQV_U5	247.1	32.3	247.1	10/10	4.3	0.00 %
LMQV_U6	288.1	34.5	288.1	10/10	7.0	0.00 %
LMQV U7	128.4	9.9	128.4	10/10	2.2	0.00 %
LMQV U8	261.1	9.3	261.1	10/10	2.4	0.00 %
LMQV U9	276.1	9.2	276.1	10/10	2.5	0.00 %
LMQV_C1	78.6	11.1	78.6	10/10	0.0	0.00 %
LMQV_C2	135.6	9.0	135.6	10/10	0.0	0.00 %
LMQV_C3	160.6	10.0	160.6	10/10	0.0	0.00 %
LMQV C4	45.2	10.5	45.3	9/10	21.1	0.20 %
LMQV C5	92.0	12.2	92.3	9/10	25.8	0.28 %
LMQV C6	94.6	12.2	94.8	9/10	24.6	0.19 %
LMQV_C7	66.4	5.6	66.4	10/10	6.3	0.00 %
LMQV_C8	132.6	5.2	132.6	10/10	10.2	0.00 %
LMQV_C9	136.7	5.7	136.7	10/10	5.2	0.00 %
$\overline{g(.,r)}$						0.04 %
$\overline{t_*(.,r)}$		17.8			8.2	
$\overline{n_{SP}(.,r)}$				9.99		
$n_{SP}(.,.)$				176/180		

Table 7
RCPSP/RCPSPR solutions (instance LMQV_U1).

		RCPSPR Solution				
		CPLEX optimal resolution		Shortest path optimal resolution		
Flow	RCPSP Solution	$\overline{h_C(x,1)}$	$\overline{t_C(x,1)}$	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$	$\overline{h_C(x,1)}$
1	19*	95	20.4	95	22.5	0.0 %
2	19	115	97.1	115	25.0	0.0 %
3	**20**	87	20.3	**87**	0.1	0.0 %
4	20	117	115.7	117	22.3	0.0 %
5	21	96	22.8	96	1.3	0.0 %
6	22	99	27.7	99	5.8	0.0 %
7	23	86	26.1	86	5.2	0.0 %
8	23	105	40.7	105	1.5	0.0 %
9	24	102	30.5	102	1.0	0.0 %
10	24	113	65.2	113	23.4	0.0 %
Average		101.5	46.6	101.5	10.6	0.0 %

Table 7 gives the set of flows randomly generated for instance LMQV_U1 (that include the optimal RCPSP flow) and provides the optimal RCPSP solution (column 2) for each flow and the optimal RCPSPR solutions provided by CPLEX and by the shortest path algorithm. The set of 10 flows introduced in Table 7 encompasses the flow of the optimal RCPSP solution. Note that the best (optimal) solution of the RCPSP has a cost equal to 19, whereas the worst one in Table 7 has a cost equal to 24. Nevertheless, the best solution found for the RCPSPR has a cost equal to 87 and was obtained with a solution of the RCPSP with a cost equal to 20 (flow number 3 in Table 7). Such results lead us to consider that high-quality solutions of the RCPSP do not lead to high-quality RCPSPR solutions, which is not really surprising since there is no reason that a quality solution for one sub-problem favors the computation of a quality solution for the whole problem. Indeed, several solutions of the RCPSP with the same makespan (for example, 23) could lead to very different solutions of the RCPSPR with a cost ranging from 86 to 105. Such behavior suggests that it would be difficult to devise a rule

that could provide quality RCPSPR solutions by considering the quality RCPSP solutions.

Table 8
Shortest path for $r = 1$ flow on each medium-scale instance.

				Shortest path optimal resolution (NL=50)	
Instances	n	n_φ	RCPSP Solution	$\overline{h_{SP}(x,1)}$	$\overline{t_{SP}(x,1)}$
LMQV_J30_U1	30	56	96	250	80.4
LMQV_J30_U2	30	57	105	276	697.0
LMQV_J30_U3	30	50	82	129	5.3
LMQV J30 C1	30	53	113	221	729.0
LMQV J30 C2	30	56	81	231	336.9
LMQV J30 C3	30	57	83	139	32.9
LMQV_J30_CC1	30	60	103	271	270.3
LMQV_J30_CC2	30	58	108	270	305.6
LMQV_J30_CC3	30	58	100	158	89.9
$\overline{t_*(.,n)}$					283.0

Table 8 reports the results obtained by the resource-constrained shortest path algorithm with a restriction of Y_i to 50 labels per node, with $r = 1$ replication per instance. In Table 8, the cost of the RCPSP solution associated with the flow is reported in column 4. Unfortunately, the best-found solution of the algorithm cannot be compared to a solution obtained with CLEX. With 30 activities, the linear formulation remains intractable due the huge numbers of both variables and constraints, including binary variables. For example, it should be noted that the instance LMQV_J30_U1 cannot be solved to optimality in 2 days of computational time using CPLEX.

For the couples instance/flow introduced in Table 8, a lower bound of the number of transport operations is provided in column n_φ. For the instance LMQV_J30_CC1, we have 30 activities plus at least 60 transport operations, representing a total of 90 operations to be scheduled, assuming that each flow is transferred by one vehicle only with one transport. The minimal number of transport operations (with a value of 60) is induced by arcs with a non-null flow.

3.3. Performance of the label management rules

To evaluate the efficiency of the algorithm, no restriction of Y_i, i.e., all labels that are required to be kept, is stored on nodes, and Table 9 provides the number of:

- labels inserted on the node due to the *INSERT_LABEL*() function;
- labels pruned on the node with *CHECK_UB_LB*() function;
- labels deleted on the node with *APPLY_DOMINANCE*() function.

All experiments were carried out on the 18 small-scale instances considering the 10 replications previously used, and the following notations are used in Table 9:

- $TNLG(x,j)$ Total number of labels generated during the replication number j;
- $\overline{TNLG(x,r)}$ Average number of labels generated during the r replications;
- $\overline{TNLP(x,r)}$ Average number of labels pruned per instance x during the r replications;
- $P_p(x,j)$ Percent of labels pruned considering $TNLG(x,j)$;
- $\overline{P_p(x,r)}$ Average percent of labels pruned with $\overline{P_p(x,r)} = Avg_{j=1,r}\left(P_p(x,j)\right)$;
- $\overline{TNLD(x,r)}$ Total number of labels dominated;
- $P_d(x,j)$ Percent of labels discarded thanks to the domination rule considering $TNLG(x,j)$;
- $\overline{P_d(x,r)}$ Average percent of labels discarded, with $\overline{P_p(x,r)} = Avg_{j=1,r}\left(P_p(x,j)\right)$;
- $\overline{P(.,.)}$ Average percent of $\overline{P_p(x,r)}$ or $\overline{P_d(x,r)}$ for all instances;
- $\overline{Avg(.,.)}$ Average number of labels generated, pruned or dominated.

Table 9
Efficiency of the label management rules with $Z_i = Y_i$ for small-scale instances.

Instances	$\overline{TNLG(x,r)}$	$\overline{TNLP(x,r)}$	$\overline{TNLD(x,r)}$	$\overline{P_p(x,r)}$	$\overline{P_d(x,r)}$
LMQV_U1	256 514.9	49 313.8	114 201.4	19.2	44.5
LMQV_U2	245 631.8	47 122.1	111 452.6	19.2	45.4
LMQV_U3	298 695.6	57 901.4	136 189.4	19.4	45.6
LMQV_U4	124 521.7	21 261.5	64 856.7	17.1	52.1
LMQV_U5	134 996.0	22 109.0	73 049.9	16.4	54.1
LMQV_U6	198 907.8	28 581.2	112 077.0	14.4	56.3
LMQV_U7	177 200.4	40 449.2	78 194.4	22.8	44.1
LMQV_U8	189 100.6	42 760.4	84 848.8	22.6	44.9
LMQV_U9	217 710.7	51 939.6	92 055.1	23.9	42.3
LMQV_C1	4.4	2.8	0.0	63.6	0.0
LMQV_C2	4.4	2.8	0.0	63.6	0.0
LMQV_C3	4.4	2.8	0.0	63.6	0.0
LMQV_C4	543 915.1	101 480.3	267 705.7	18.7	49.2
LMQV_C5	589 227.4	107 571.8	281 184.5	18.3	47.7
LMQV_C6	580 233.6	107 110.3	280 699.6	18.5	48.4
LMQV_C7	230 693.0	26 232.9	167 281.8	11.4	72.5
LMQV_C8	294 602.6	35 820.0	203 370.0	12.2	69.0
LMQV_C9	273 935.0	33 845.2	187 245.9	12.4	68.4
$\overline{Avg(.,.)}$	241 994.4	42 972.6	125 255.8		
$\overline{P(.,.)}$				25.4	43.6

The following remarks hold:

- 25% of the labels generated are pruned (thanks to the function $CHECK\_UB\_LB(L, LB, UB)$);
- 43% of the labels generated are deleted (thanks to the dominance rule with $DOMINATE(i,j)$ and $APPLY\_DOMINANCE(L, Z)$.

These results lead us to consider that the dominance rule we introduced is highly efficient and defines the cornerstone of the shortest path algorithm.

3.4. Remarks

The results first confirm the algorithm efficiency from a computational point of view and prove that the algorithm can be used in practice and could be integrated into an iterative metaheuristic-based search on an indirect representation of solutions (Cheng et al., 1996) (Prins, 2002). The resource-constrained shortest path algorithm can be used to determine an evaluation function that defines the optimal RCPSPR solution for one flow. From this point of view, the computational efficiency of the method is crucial. Intensive numerical experiments have shown that the algorithm can be adapted depending on the context, and should make it possible to find quality RCPSPR solutions within an acceptable computational time. An experiment based on 18 flows revealed that the computational time can be reduced to 15 seconds with 500 labels per node, and to 0.4 seconds with 50 labels per node, providing the optimal RCPSPR solution in 67% of the flows (the average deviation to the optimal solution is less than 0.9%). Because the evaluation function that makes it possible to associate a flow with a solution can be a part of a global optimization process, the capacity of the algorithm to ensure quasi-optimal evaluation in a very short computational time is a significant highlight of the algorithm.

4. Concluding remarks

A new resource-constrained shortest path algorithm is introduced to define a RCPSPR solution from one RCPSP flow solution. The RCPSPR is a new integrated problem dealing with scheduling and routing based on the RCPSP to which routing constraints with a heterogeneous fleet of vehicles with limited capacities have been added.

Our contribution concerns: (1) the label definition that encompasses both system state and resources; (2) the dominance rule; (3) the propagation rule.
The algorithm ensures an efficient algorithmic solution to deal with a proper coordination between scheduling and routing problems since it points the way towards a definition of an iterative search process based on an indirect representation by an efficient computation of RCPSP solutions from a flow of the RCPSP. It should be noted that the resource-constrained shortest path algorithm we have introduced is a new way to solve arc routing problems by shortest path computation.

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. An efficient new heuristic for the hoist scheduling problem. Computers & Operations Research 2016; 67: 184-192.

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

Alvarez-Valdés R., Tamarit J.M. The project scheduling prolyhedron: Dimension, facets and lifting theorems. European Journal of Operational Research 1993; 67: 204-220.

Artigues C., Michelon P., Reusser S. Insertion for static and dynamic RCPSP. European Journal of Operational Research 2003; 149: 249-67.

Caumond A., Lacomme P., Moukrim A., Tchernev N. An MILP for scheduling problems in a FMS with one vehicle. European Journal of Operational Research 2009; 199(3): 706-722.

Cheng, R., Gen, M. Tsujimura, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms – I representation. Computers and Industrial Engineering 1996; 30: 983–997.

Chtourou, S., Manier, M.-A. Loukil, T. A hybrid algorithm for the cyclic hoist

scheduling problem with two transportation resources. Computers & Industrial Engineering 2013; 65(3): 426-437.

Cordeau, J.-F., Laporte, G. A tabu search heuristic for the static multi-vehicle dial-a ride problem. Transportation Research Part B Methodology 2003; 37: 579–594.

Dauzère-Pérès S., Lasserre J.B. 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. 1995.

Dongarra, J. Performance of various computers using standard linear equations software. Report CS-89-85, University of Manchester. 2014.

Firat M., Woeginger G.J. Analysis of the dial-a-ride problem of Hunsaker and Savelsbergh. Operations Research Letters 2011; 239: 32–35.

Golden B.L., Wong, R.T. Capacitated arc routing problems. Networks 1981; 11: 305–315.

Honglin Q., Sujing W., Qiang Xu. A new method of cyclic hoist scheduling for multi-recipe and multi-stage material handling processes. Computers & Chemical Engineering 2016; 90: 171-187.

Knust S. Shop-Scheduling Problems with Transportation. Ph.D. thesis. Fachbereich Mathematik/ Informatik, Universität Osnabrück. 1999.

Kwan, M-K. Graphic programming using odd or even points. Chinese Mathematics 1962; 1: 237–277.

Lacomme P., Prins C., Ramdane-Chérif W. Evolutionary algorithms for periodic arc routing problems. European Journal of Operational Research. 2005; 165: 535-553.

Lacomme P., Larabi M., Tchernev N. A disjunctive graph for the job-shop with several robots. In: MISTA conference, Paris, France. 2007.

Lacomme P., Larabi M., Tchernev N. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles, International Journal of Production Economics 2013; 143(1): 24-34.

Lacomme P., Moukrim A., Quilliot A., Vinot, M. 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. 2017.

Moons S., K. Ramazkezrs, A. Caris and Y. Arda. Integrated production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. Computers and Industrial Engineering. 2017; 104: 224-245.

Orloff C.S. A fundamental problem in vehicle routing. Networks 1974; 4: 35–64.

Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 2004; 31(12): 1985-2002.

Pritsker A., Watters L. A zero-one programming approach to scheduling with limited resources. The RAND Corporation, RM-5561- PR. 1968.

Pritsker A., Watters L., Wolfe P. Multi-project scheduling with limited resources: a zero-one programming approach. Management Science 1969; 16: 93-108.

Quilliot A., Toussaint H. Resource Constrained Project Scheduling with Transportation Delays, IFAC Proceedings Volumes, 2012; 45(6): 1481-1486.

Roy B., Sussmann B. Les problèmes d'ordonnancement avec contraintes disjonctives. Note DS 1964; 9 bis, SEMA, Paris.

Saglam Ü., Banerjee A. Integrated multiproduct batch production and truck shipment scheduling under different shipping policies. Omega 2017, http://dx.doi.org/10.1016/j.Omega.2017.01.007.

Valls V., Ballestín F., Quintanilla S. Justification and RCPSP: A technique that pays. European Journal of Operational Research, 2005, 165(2): 375-386

Zhang G., Nishi T., Turner S.D.O, Oga K., Li X. An integrated strategy for a production planning and warehouse layout problem: Modeling and solution approaches. Omega 2016; 68: 85-94.

Zhang Q., Manier H., Manier M.-A. A Modified Disjunctive Graph for Job-Shop Scheduling Problems with Bounded Processing Times and Transportation Constraints. IFAC Proceedings 2012a; 45(6): 1377-1382.

Zhang Q., Manier H., Manier M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computers & Operations Research 2012b; 39(7): 1713-1723.