Maximum Order Tree Algorithm for Optimal Scheduling of Product Distribution Lines

S. D. Mokashi and A. C. Kokossis
Dept. of Process Integration, UMIST, Manchester, U.K.

Research on scheduling and planning in the chemical engineering community subscribes to one of two schools of thought. A general-purpose optimization approach resorts to using conventional mathematical programming techniques on generic models of a scheduling problem, which has a limitation in its application to large-scale industrial problems in terms of the computational time involved. The other extreme of heuristic methods lacks guarantees on the quality of the solution. A philosophy is proposed of contextual optimization that exploits problem-specific knowledge to develop efficient algorithms. This concept is applied to a delivery scheduling problem to generate a tailored graph-based method called the maximum order tree algorithm, which reduces the CPU time dramatically compared to conventional methods without compromising on the quality of the solution. When applied to a single-site distribution case study, it resulted in savings of over a quarter of a million dollars per year over the existing heuristic-rule-based system.

Introduction

The area of scheduling and planning in the chemical engineering community has witnessed numerous attempts towards optimization of the supply chain. These efforts can be classified under two major philosophies: a heuristic approach in the interest of getting solutions in real time and a so-called rigorous general-purpose optimization endeavor. The demerit of the former lies obviously in having no guarantees about the quality of the solution, thereby undermining the benefits from potential savings. The latter philosophy has a fundamental limitation in terms of its application to scheduling problems of industrial interest.

The generic approach aims at building general purpose models reflecting the operations in or related to a chemical plant. The State Task Network models (Kondili et al., 1993), for example, claim to provide a rigorous framework for solving production scheduling problems with intermediate storage. Although this is a good modeling tool for batch operations, the corresponding optimization effort could be a Herculean task. Notwithstanding the efforts at tightening problem constraints to relieve the strain on the MILP solver, there are no (Shah et al., 1993) rigorous mathematical guarantees as to what sub-parts of the optimization problem ensure integrality for the corresponding relaxed LPs. This forces the end user to impose artificial margins of optimality or to try and fix some of the integer variables in order to wheedle out so-called optimal solutions in real time. The situation gets worse when it comes to the application of these approaches to real life industrial problems. The discrete time models proposed in the above approach have to use unreasonably long time intervals to tame the size of the model (Wilkinson et al., 1993).

If the generic approach philosophy is carried over to other areas of the supply chain, like product distribution, it could lead to disastrous results on industrial problems. An MILP approach is illustrated in this article to highlight the need for a surrogate school of thought. Delivery problems can very quickly run out of proportion when formulated as generic models and solved using general purpose solvers, since the number of resources in delivery problems can be far more than those in their production counterparts. Also, these problems may have complex changeover policies which are impossible to model (Shah et al., 1993) or difficult to handle using the existing modeling techniques for production scheduling.

Correspondence concerning this article should be addressed to A. C. Kokossis at this current address: Dept. of Chemical and Process Engineering, University of Surrey, Guildford, GU2 5XH, Surrey, U.K.

This situation will be further aggravated when these generic techniques are applied to design problems where the capacity and number of resources become optimization variables.

Since optimization of product distribution forms an important component of the supply chain, it is worthwhile to consider what other disciplines have to offer in terms of optimization technology. Operations research literature is replete with work on variations of conventional transportation type (Junginger, 1995; Rajendra Prasad et al., 1993) problems as also on vehicle routing problems (Laporte, 1992). Apart from the more generic class, work has also been done on developing heuristics for specific problems of practical interest like allocation of trucks to customer orders (Ronen, 1992; Brown and Ronen, 1997). A significant feature of these efforts has been the existence of tailor-made algorithms, in cases where general-purpose solution techniques may fail, which address specific problems as opposed to setting up ambitious tasks.

The delivery problem related to a process plant does not necessarily fall into the category of the celebrated transportation or trans-shipment problems and, hence, is not easily amenable to optimization using these formulations. Some of the heuristics in operations research literature come close to addressing this problem, but cannot address features like product switching constraints which are so relevant to the process industry. In the face of this dearth of robust optimization technology for scheduling problems of industrial relevance, the logistics engineer has to resort to heuristics based on rules developed from experience or arbitrary biases.

This article proposes a tailored approach for a delivery problem related to a typical production facility. For the purpose of this article, it is assumed that production and delivery scheduling are not integrated. The limitations of a general purpose mathematical framework are highlighted first to establish the motivation of an alternative tool. These limitations are removed in the new approach by exploiting problem-specific information. A graph-based representation is proposed coupled with a tailored research over this graph. This approach is applied to an industrial case study, and its benefits over an existing rule based system are demonstrated. Finally, conditions of industrial relevance under which the optimality of this algorithm can be guaranteed are discussed.

Delivery Scheduling Problem

The delivery scheduling problem alluded to in the introduction consists of the optimal allocation of the distribution resources to satisfy various orders (demands) from customers. The distribution resources are mainly carriers of different types which could be lorries, trucks, rail containers, ships, and so on. Carriers happen to be the major cost component of interest. Their cost includes fuel for transport, manpower involved in the transportation, and other miscellaneous constituents attached to the delivery. The optimization objective is to allocate these resources to satisfy the orders from different customers. Customer orders have various specifications like the amount of product, due date by which the supply should be received, product grade, and packing type of product. See Table 1 for a sample list of orders. For the purpose of problem formulation, any order o can be fully characterized as

$$o \equiv \{dd_o, p_o, Q_o\} \tag{1}$$

Table 1. Sample List of Orders

Order	Customer Code	Packing Type	Product Grade	Due Date
1	15	Bags	2	950331
2	4	Containers	23	950331
3	22	Tins	12	950402
⋯	⋯	⋯	⋯	⋯
n_{cus}	8	Barrels	10	950406

where p_o is the product/grade and Q_o is the volume/amount of p_o ordered.

More specific order specifications can also be incorporated in the formulation, if desired. Also, dd_o may be the departure time from the source location or the arrival time at the destination. The arrival/departure times may not be particular times, but a time interval. For the purpose of developing the representation, the above order representation can be used in the interest of simplicity. The "Extensions to the Framework" section includes a discussion on the extension of this to incorporate time windows and its impact on algorithmic performance.

The delivery cost for a particular order depends on the category of carrier resources allocated to it. The various carrier categories may have different delivery costs for the same order due to one or more of the following reasons:

(1) The categories may have different capacities, and, hence, the higher capacity carriers may have lower delivery cost per unit of product delivered.

(2) The categories may be of different types like those running on road and rail.

(3) The carrier type may be externally hired, in which case the costs are higher in compensation for greater flexibility. Since external carriers need not be scheduled by the company, they can be hired when required.

Each carrier has a date of arrival from its previous trip. The previous trip refers to the delivery made before the start of the current scheduling horizon. The carriers have to be allocated to the orders such that the utilization of any carrier at any given time should not conflict with any other order, neither with those belonging to the current time horizon, nor with those from the previous one.

Each carrier carries one product at a time to one customer and returns empty. This policy corresponds to orders delivered in bulk, as opposed to small packings. Between two subsequent trips, there are restrictions on product switching. Certain products cannot be loaded immediately after some others have been delivered on the last trip (see Table 2). This is to prevent contamination of products especially in the case of those which have an end-use in highly sensitive areas like food, medicine, and so on. The product switching information can be represented with the $nP \times nP$ matrix **SP**. Apart

Table 2. Cleaning Constraints for Product Switching

Product	1	2	⋯	nP
1	√	√	⋯	X
2	√	√	⋯	√
⋯	⋯	⋯	⋯	⋯
p	X	√	⋯	√

from forbidden switching, products that can be switched involve cleaning or changeover costs. Since these are negligible compared to the delivery cost, these are ignored in the model.

Although the scope of the problem definition is set with the above limitations, the theory can be extended to the following cases:

(1) Instances where two or more neighboring customers/products are delivered by the same carrier. The application discussed in a later subsection in the article includes full trailers that can accommodate two customer orders with different products. This article, however, does not not address the routing issue of a large number of customers since it is qualified to a scheduling problem and not focused on routing or traveling salesman problems.

(2) Cases where changeover time and costs are important (see first subsection in "Extensions to the Framework" section).

Problem statement

The delivery problem can now be summarized as follows.

Given. (1) Carriers of nC categories with different product carrying capacities and different delivery costs $DC_{o,i}$. These categories can be arranged in increasing order of delivery cost per unit of product

$$UDC_{o,i} < UDC_{o,i'} \qquad \forall i < i', \quad \forall o \in O \tag{2}$$

where

$$UDC_{o,i} \overset{def}{=} \frac{DC_{o,i}}{cap_i} \qquad i = 1, \ldots, nC+1, \quad \forall o \in O \tag{3}$$

Note: The delivery cost is not given as per unit volume, but as a function of the carrier type and the distance of the trip. The above definition of unit cost $UDC_{o,i}$ is introduced for the propose of explaining the theory behind the algorithm.

(2) C_{nC+1} corresponds to the category of carriers hired externally in case the delivery requirements cannot be met from the set of carriers $C_1, C_2, \ldots, C_{nC}$ owned by the company.

(3) nc_i carriers in each category C_i.

(4) Arrival time $da_{i,j}$ and product carried in the previous trip $pp_{i,j}$, by each carrier $c_{i,j}$.

(5) A set O of n customer orders o (defined by Eq. 1), each constituting of a customer, product grade, amount of product ordered and due date by which the customer requires the product to be delivered.

(6) nP product grades and a $nP \times nP$ product switching matrix **SP**, where

$$SP_{p,p'} = \begin{cases} 1 & \text{if } p \rightarrow p' \\ 0 & \text{if } p \nrightarrow p' \end{cases} \tag{4}$$

Objective. Schedule the orders $o \in O$ on the carriers $c_{i,j}$ $(i \le nC)$ and determine the requirement of external carriers C_{nC+1} that will minimize the total delivery cost and satisfy all the constraints in question.

Optimization Challenges

As an initial attempt, the problem was formulated in a mathematical programming framework and solved using a branch and bound solver. The formulation resulted in a MILP. Let the binary variables $y_{i,j,o}$ represent the allocation of order o to carrier $c_{i,j}$ and the set of integer variables n_o^{ext} represent the number of external carriers required.

Objective. The objective function is the total delivery cost, which is the sum of the delivery costs of individual orders on the carriers allotted to them

$$TDC = \sum_{i=1}^{nC-1} \sum_{j=1}^{nc_i} \sum_{o \in O_i} y_{i,j,o} DC_{o,i} + \sum_{o \in O} n_o^{\text{ext}} DC_{o,nC} \tag{5}$$

Unique Utilization of Resources. Each carrier resource should be allotted to only one order at any time. Hence, if it is used for some order o, it cannot be used for some other order o' between the time $dd_{o,i}$ and $dd_{o,i} + ret_{o,i}$ ($ret_{o,i}$ is the journey time for delivery order o and returning to the source using carrier type C_i) and $dd_{o,i}$ is the corresponding departure time from the source

$$y_{i,j,o} + y_{i,j,o'} \le 1 \tag{6}$$
$$\forall (o,o') \in OO',\ j = 1, \ldots, nc_i,\ i = 1, \ldots, nC$$
$$OO' = \{(o,o') | dd_{o,i} \le dd_{o'} \le dd_{o,i} + ret_{o,i}\}$$

If the departure time of o' is before the arrival time of the vehicle from its last trip on previous scheduling horizon, then order o must belong to the previous time horizon ($o \equiv \hat{o}$); therefore, $y_{i,j,o} \equiv y_{i,j,\hat{o}} = 1$. Then, Eq. 6 simplifies to

$$y_{i,j,o'} = 0 \tag{7}$$
$$\forall o' \in O',\ j = 1, \ldots, nc_i,\ i = 1, \ldots, nC$$
$$O' = \{o' | dd_{o'} \le da_{i,j}\}$$

Product Switching Constraints. In the case of forbidden product switching, there must be at least one other order o'' delivered between a forbidden switching pair o,o' corresponding to $SP_{p_o,p'_o} = 0$, so that the delivery schedule $o \rightarrow o'' \rightarrow o'$ is feasible, although $o \rightarrow o'$ is infeasible

$$y_{i,j,o} + y_{i,j,o'} - \sum_{o'' \in O''} y_{i,j,o''} \le 1 \tag{8}$$

where

$$O'' = \{o'' | dd_{o,i} + ret_{o,i} \le dd_{o''} \le dd_{o'',i} + ret_{o'',i} \le dd_{o'}, SP_{p_o,p''_o} = SP_{p''_o,p'_o} = 1\},\ \forall o \in O,\ \forall o' \in O' = \{o' | SP_{p_o,p'_o} = 0\}\ i = 1, \ldots, nC,\ j = 1, \ldots, nc_i.$$

In case o happens to be an order from the previous time horizon, then since $y_{i,j,o} \equiv y_{i,j,\hat{o}} = 1$ Eq. 9 reduces to

$$y_{i,j,\hat{o}'} - \sum_{o'' \in O''} y_{i,j,o''} \le 0$$

where

$$O'' = \left\{ o'' | da_{i,j} \le dd_{o''} \le dd_{o'',i} + ret_{o'',i} \le dd_{o'}, SP_{p_{i,j},p_{o''}} = SP_{p_{o''},p'_o} = 1 \right\},$$

$$\forall i \le nC,\ j \le nc_i\ \forall o' \in O' = \left\{ o' | SP_{p_{i,j},p_{o'}} = 0 \right\}$$

Fulfillment of Order Quantities. The carrier capacities allotted to an order must be greater than or equal to the amount of product ordered

$$\sum_{i=1}^{nC} \sum_{j=1}^{nc_i} y_{i,j,o} \cdot \mathrm{cap}_i + n_o^{\mathrm{ext}} \cdot \mathrm{cap}_{nC+1} \ge Q_o \quad \forall o \in O \quad (10)$$

Equation 10 may not be a strict equality because the quantities Q_o may not be integral multiples of capacities cap_i.

The MILP formulation was done in GAMS using OSL as the branch and bound solver. It was relatively straightforward to obtain the optimal schedule when the number of carriers and orders was small. Table 5 will show CPU times for small problems with $nO \le 15$ and $\sum_i nc_i = 70$, $nC = 3$. However, the number of variables and constraints and, hence, the memory requirement increases dramatically with an increase in the number of order and carriers. For larger sized problems ($nO \ge 20$, $\sum_i nc_i \ge 70$), OSL was unable to obtain any solution with reasonable relative integrality gap. Also, for a scheduling horizon above 6 days, it was not possible to launch the solver. Apart from the massive size of the model itself, the other fundamental problem with this approach is the large integrality gap.

The use of solvers adept at handling sparse matrices can be perceived as a solution to counter the massive size. Any effort in this direction would not undermine the problem caused by the integrality gap.

Another effort to facilitate optimization could be in terms of employing some model tightening techniques to reduce the integrality gap. A generic approach would be to use a cutting planes method (Carroe and Tind, 1997). The benefits of doing this are far from obvious since the task of identifying cutting planes iteratively could involve its own overheads, like its branch and bound counterpart. Another endeavor could be toward writing constraints in a tighter form *a priori*. This would be similar to the work of Shah et al. (1993) in which the authors suggest backward propagation constraints. Applying this idea to the resource utility constraints 6 and 7 results in the following equations

$$y_{i,j,o} + \sum_{o' \in O'} y_{i,j,o'} \le 1 \qquad O' = \left\{ o' | dd_{o',i} \le dd_{o,i} \le dd_{o',i} + ret_{o',i} \right\}, \quad (11)$$

$$\forall o \in O,\ j = 1, \ldots, nc_i,\ i = 1, \ldots, nC$$

$$\sum_{o' \in O'} y_{i,j,o'} \le 0 \qquad O' = \left\{ o' | dd_{o',i} \le da_{i,j} \right\}, \quad (12)$$

$$j = 1, \ldots, nc_i,\ i = 1, \ldots, nC$$

Each of the above constraints now contain more orders o (compared to the naive constraints 6 and 7) which conflict with each other in terms of their delivery times. Admittedly, this may result in effective tightening in some specific cases, but in the context of the more general changeover policies relevant to delivery problems, they have their own limitations. Further, these attempts lack a rigorous *a priori* analysis so as to justify their application in a generic sense although they may be applicable to specific changeover policies considered in their article.

A reduction in computational time could also be achieved by using the trivial cutting plane $Q_o^{\min}$ on the order fulfillment constraint. $Q_o^{\min}$ is the minimum amount realized through a linear combination of carrier capacities.

Constraint 10 can be replaced by the following

$$\sum_{i=1}^{nC} \sum_{j=1}^{nc_i} y_{i,j,o} \mathrm{cap}_i + n_o^{\mathrm{ext}} \mathrm{cap}_{nC+1} \ge Q_o^{\min} \quad \forall o \in O \quad (13)$$

However, this reduction is significant only if most of the quantities are fractional amounts of the capacities and $Q_n^{\min}$ is significantly different from Q_o.

A variable disaggregation policy can also be borrowed from the celebrated (Balas, 1998; Pinto and Grossmann, 1997) disjunctive programming approach to provide a better replacement of Constraint 13. Such an effort is not enough on its own to address the looseness in the changeover policy constraints.

Since it is clear that generic approaches do not provide a panacea to the fundamental problems chronic to scheduling problems, it seems to be advisable to use less ambitious tailored methods that exploit problem specific information and use representations that are amicable with their constraints and peculiar structure. The advantages of using problem specific or contextual information to facilitate optimization have already been demonstrated (Mokashi and Kokossis, 1997). This article provides the details behind the application of contextual optimization to the delivery problem. The failure of a generic method provides the motivation to evolve a specific algorithm with a representation requiring modest memory and a search that exploits the important trade-offs.

Development of Maximum Order Tree Algorithm

Outline

The Maximum Order Tree (MOT) method is based on a graph representation of customer orders. This representation will be introduced first, followed by a discussion of the important trade-offs in the problem. This will lead to the evolution of a quantity termed the resource potential, and in a systematic sequence finally to the application of the search criterion to the graph of orders. Important procedural elements of the algorithm will be discussed next. Finally, the algorithm will be presented in a formal manner.

Graph representation of orders

The customer orders have a precedence antecedent relationship with each other depending on the position of their delivery time with respect to the start of the scheduling horizon and other orders. This relationship can be very neatly

Table 3. Sample Order Data

Order	dd_o	ret_o	pro
o_1	0	1	2
o_2	0	2	3
o_3	2	1	1
o_4	2	2	4
o_5	3	2	2
o_6	4	1	3

captured by a graph representation. Graph $G=(V,E)$ is a set of vertices V and edges E representing the connectivity between the vertices (Biggs and Ronen, 1997). The set of orders O can be represented as the set of vertices V. The precedence-antecedent relationship between orders $o \in O$ will then correspond to the set of directed edges E.

Considering the resource utilization constraint (Eq. 6), the edge set E can be defined as a set of ordered pair of orders satisfying this constraint

$$E=\{(o,o')|o,o' \in O,\ dd_{o'} \geq dd_o + ret_o\} \qquad (14)$$

Table 3 shows a sample set of orders, and Figure 1 illustrates the corresponding graph representation.

If the product switching constraints (Eq. 9) are to be incorporated in the graph, then a more generic definition of the edge set E can be used

$$E=\{(o,o')|o,o' \in O,\ dd_{o'} \geq dd_o + ret_o,\ SP_{p_o,p_{o'}}=1\} \qquad (15)$$

Equation 16 shows the SP matrix for the product involved in the sample set of orders in Table 3. Imposing the switching constraints onto the graph of Figure 1 produces the graph in Figure 2

$$\boldsymbol{SP}=\begin{bmatrix}1&1&0&1\\1&1&0&1\\0&1&1&1\\1&1&1&1\end{bmatrix} \qquad (16)$$

Referring to Figure 2, connections between orders o_1 and o_5 are of no consequence in the presence of the connections between $o_1 \rightarrow o_3$ and $o_3 \rightarrow o_5$ since o_1 and o_5 are connected in an indirect sense through o_3 even in the absence of the edge (o_1,o_5). The edge (o_1, o_5) is therefore redundant in the presence of o_3. Two types of connections between vertices need to be defined in the light of their importance in the algorithm.

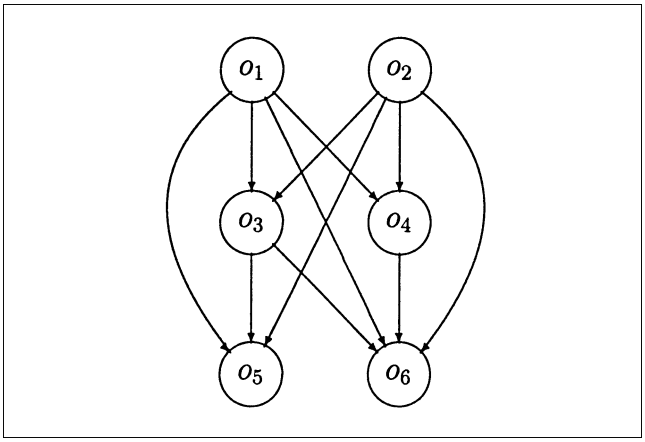

Figure 1. Order graph.

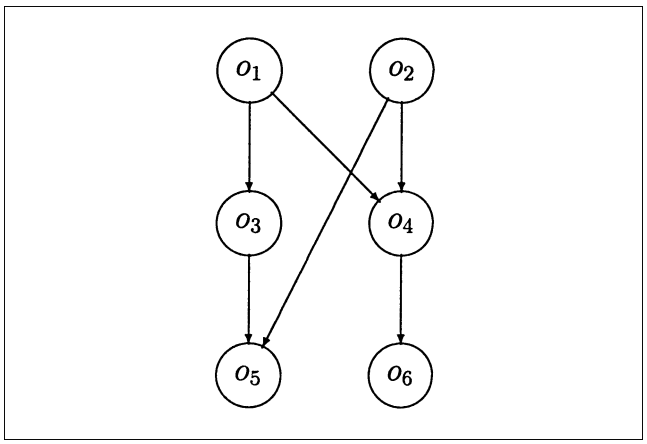

Figure 2. Order graph without redundant edges.

Definition 1 (Indirect Connection). *A pair of orders o,o′ are said to be indirectly connected if* $\exists(o,o_1), (o_1,o_2), (o_2,o_3), \ldots,(o_n,o') \in E,\ n \geq 1.$ $(o,o') \in E^{id}(G) \equiv$ *set of indirect connections of graph G.*

Definition 2 (Direct Connection). *A pair of orders o,o′ are said to be directly connected if* $\exists(o,o') \in E$ *and* $(o,o') \notin E^{id}(G)$.

Resource potential

The order graph discussed in the previous section establishes a framework to perform the optimization search. The search procedure itself will be based on a quantity termed as the resource potential, which captured important trade-offs in the problem. This search does not use any of the conventional optimization solvers but is a problem specific algorithm.

Consider a set O of orders, such that the edge set $E=\emptyset$. Since there are no connections between any vertices, there is no precedence-antecedent relationship in this case. Hence, any carrier can be used to deliver exactly one order from set O. Now, consider allocation of any carrier $c_{i,j} \in C_i$ to one of the orders from set O. If an order is not selected on category C_i, then it could be possibly selected on the next category C_{i+1} of higher delivery cost per unit volume, assuming that the carrier categories are arranged in ascending order of cost per unit amount of product. This happens to be the dominant trade-off of this problem. On a stand-alone basis, if a choice between the n orders in set O is to be made, then the order with the maximum value of

$$RP_{o,i} \overset{def}{=} UDC_{o,ext} - UDC_{o,i} = \frac{DC_{o,nC+1}}{\text{cap}^{ext}} - \frac{DC_{o,i}}{\text{cap}_i} \qquad (17)$$

should be selected as this will minimize the loss on transferring the remaining orders to the higher cost carriers.

Table 4. Calculation of Resource Potential

Order	$DC_{o,i}$	$DC_{o,\text{ext}}$	$RP_{o,i}$
o_1	2.8	1.6	0.2
o_2	4.0	2.5	0.5
o_3	2.4	1.4	0.2
o_4	6.4	3.9	**0.7**
o_5	4.8	2.9	0.5
o_6	3.6	2.1	0.3
cap_i	2	1	

The idea of the resource potential can be illustrated with the example in Table 4.

Order o_4 is the one with the maximum value of resource potential w.r.t. carrier type C_i and hence must be given priority among the six orders listed while allotting any carrier $c_{i,j}$.

The resource potential by itself is applicable only for the stand-alone scenario discussed above. However, it can be coupled with the graph representation of customer orders, to serve as a handy search criterion.

Application of resource potential to order graph

The resource potential criterion defined for a graph $G = (0,\emptyset)$ can be extended to any graph $G = (O, E)$. The resource potential $RP_{o,i}$ can be attached as a weight to the vertex o to obtain a weighted graph for carrier type C_i. This is best illustrated by revisiting the sample list of orders.

A weighted version of the graph in Figure 2 is as shown in Figure 3.

The graph $G = (O,E)$ where

$$O = \{o_1, o_2, \ldots, o_6\}$$

and

$$E = \{(o_1,o_3),(o_1,o_4),(o_2,o_4),(o_2,o_5),(o_3,o_5),(o_4,o_6)\}$$

can be transformed to a graph $G' = (O',\emptyset)$, where

$$O' \equiv \{(o_1,o_3,o_5),(o_1,o_4,o_6),(o_2,o_4,o_6),(o_2,o_5)\}$$

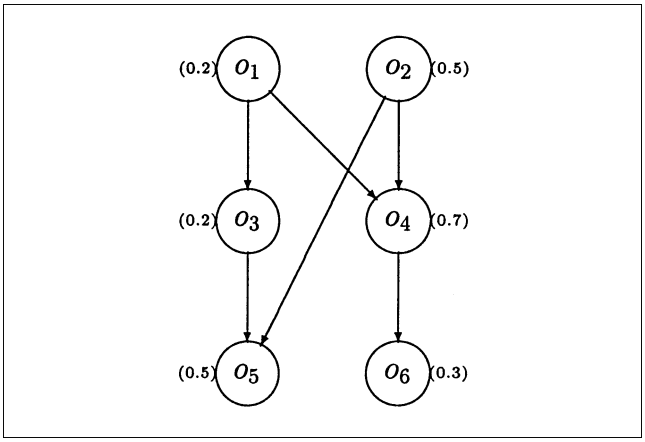

Figure 3. Order graph with resource potential.

Notice that the graph G' could also have included other options like (o_1,o_3) and (o_3,o_5). However, these are subsets of the already included order combinations and, hence, redundant in terms of finding the vertex with the maximum value of resource potential.

The corresponding resource potentials for elements of O' will then be

$$\begin{bmatrix} RP_{o_1,1} + RP_{o_3,1} + RP_{o_5,1} \\ RP_{o_1,1} + RP_{o_4,1} + RP_{o_6,1} \\ RP_{o_2,1} + RP_{o_4,1} + RP_{o_6,1} \\ RP_{o_2,1} + RP_{o_5,1} \end{bmatrix} = \begin{bmatrix} 0.9 \\ 1.2 \\ \mathbf{1.5} \\ 1.0 \end{bmatrix}$$

Now applying the resource potential criterion it is easy to see that the tree $T^* = (o_2, o_4, o_6)$ is the one with the maximum resource potential. Hence, formally defining the criterion for a general graph $G = (O,E)$, the tree T^* with the maximum value of RP_T should be selected

$$RP_{T^*} = \max_{T \in G} \sum_{o \in T} RP_{o,i} \tag{18}$$

For the purpose of this discussion, the single branch tree can be formally defined as follows.

Definition 3 (Single Branch Tree T). *A single branch tree is defined as a directed tree in which each vertex (order) of the tree is connected to one other vertex in the tree, except the bottom-most vertex.*

The enumeration of order combinations tends to leave an impression that the algorithm will use exhaustive enumeration and, hence, be computationally expensive. The arguments outlined in this section are more for a conceptual understanding of the representation, rather than the actual implementation itself. The maximum order tree can be found using a customized branch and bound search.

Procedure elements of the MOT algorithm

The MOT algorithm essentially puts the ideas of order graph and resource potential in a sequential procedure that consists of the following elements:

(1) Development of order graph $G_{i,j}$ for each carrier $c_{i,j}$.

(2) Search of the maximum order tree T^* belonging to this graph $G_{i,j}$.

(3) Updating of connections by eliminating orders that have been satisfied and establishing connections across them.

The algorithm repeats this procedure starting from the cheapest category and the earliest available carrier in each category.

The development of order graph was discussed earlier and the resource potential to be associated with the vertices was explained also. The search can be exhaustive. The customized branch and bound algorithm discussed later can be used. The following definitions need to be introduced for their use in the MOT algorithm while updating connections across exhausted orders/vertices.

Definition 4 (Exhausted Order/Vertex). *An Order/Vertex o is said to be exhausted with respect to selection on a carrier* $c_{\text{i,j}}$ *if the capacity assigned* Q_o^{assign} *for o satisfies the required demand* Q_o.

$$\sum_{i'=1}^{i-1}\sum_{j'=1}^{nc_{i'}} \mathrm{cap}(o,T_{i',j'}) + \sum_{j'=1}^{j-1} \mathrm{cap}(o,T_{i,j'}) = Q_o^{\mathrm{assign}} \geq Q_o \quad (19)$$

where,

$$\mathrm{cap}(o,T_{i,j}) = \begin{cases} \mathrm{cap}_i & \forall o \in T_{i,j}, \\ 0 & \text{otherwise}. \end{cases} \quad (20)$$

V_i^{ex} is the set of exhausted vertices on carrier categories $C_{i'}$, $i' \leq i$, and $V_{i,j}^{ex}$ is the set of exhausted vertices on category $c_{i,j}$.

Definition 5 (Child/Parent Vertices). *For any directed edge $(o,o') \in G$ the vertex o is termed the parent of o' and o' is called the child of o.*

$$Pa(o) = \{o' | (o',o) \in E\} \quad (21)$$

$$Ch(o) = \{o' | (o,o') \in E\} \quad (22)$$

For the purposes of the MOT algorithm, the edge set E in Definition 5 is replaced by E^d to avoid redundant searches.

Now, if any order o is exhausted w.r.t. $c_{i,j}$ then any two vertices o^p, o^c which were above ($o^p \rightarrow o$) and below ($o \rightarrow o^c$) are connected ($o^p \rightarrow o^c$) if this connection exists in the original graph G_i.

MOT algorithm

The MOT algorithm can now be formalized, utilizing the concepts discussed in the previous sections:

(1) Arrange carrier types in increasing order of delivery cost per unit product $\{C_1, C_2, \ldots, C_{nC}\}$, $UDC_{o,i} < UDC_{o,i'} \forall i < i'$, $\forall o \in O$.

(2) Iterate steps 3 and 4 for categories C_i, $i = 1, \ldots, nC$. Go to step 7.

(3) Initialize for category C_i, start at carrier $c_{i,1}$ $j = 1$.

(a) $Ub(o) = \infty$ $\forall o$, $Ub = \infty$.

(b) $Lb(o) = 0$ $\forall o$, $Lb = 0$.

(c) Generate G_i using relation 15.

(i) If $i = 1$ generate G_i from O_1.

(ii) If $i > 1$ generate G_i from $O_i = (O - V_i^{ex}) \cap O_i$ removing exhausted vertices on previous categories.

(d) Initialize graph G for the first carrier $c_{i,1}$. Let $G = G_{i,j}$.

(4) Iterate steps 5 and 6 for carriers $c_{i,j}$ $j = 1, \ldots, nc_i$.

(5) Initialize graph G for carrier $c_{i,j}$, $j > 1$.

(a) $Lb(o) = 0$ $\forall o$, $Lb = 0$.

(b) $Ub_{ij,}(o) = Ub_{i,j-1}(o)$ (see Lemma 1 in the next section).

(c) $G = G_{i,j} - V_{j-1}^{ex}$

(d) Establish connections across exhausted vertices $v^{\mathrm{ex}} \in V_{j-1}^{ex} \subseteq T_{i,j-1}$.

$$E^d(G) = E^d(G) \cup \left\{(o^p,o^c) | o^p \in Pa(o),\ o^c \in Ch(o),\ (o^p,o^e) \in E^d(G_{i,j}),\ o^p,o^c \in G,\ o \in V_{j-1}^{ex}\right\}$$

(6) Find the single branch tree $T_{i,j} \in G$ starting with $\hat{o}_{i,j}$ with the maximum value of resource potential *sumRP*.

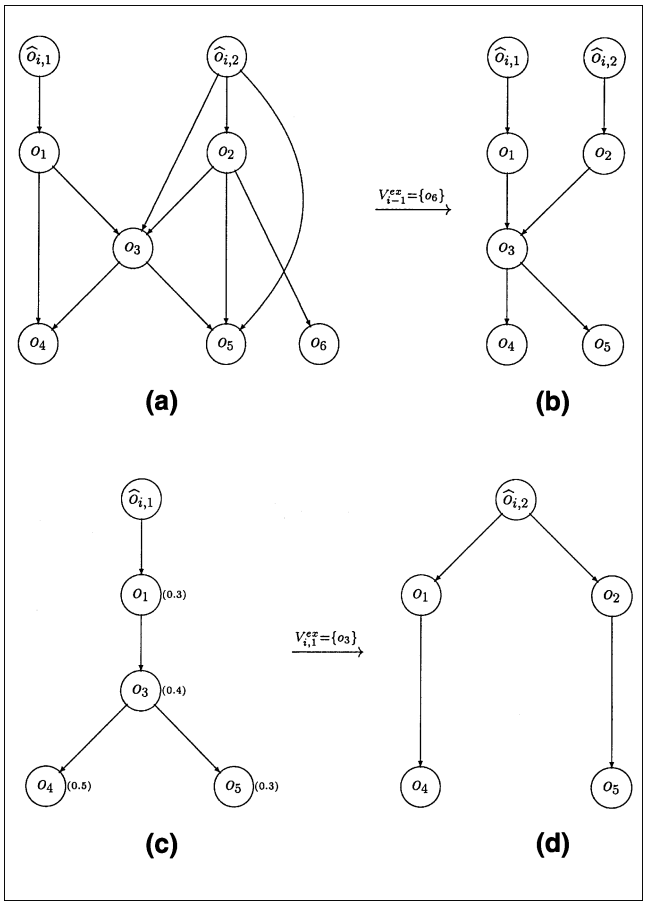

Figure 4. MOT algorithm.

(a) Original graph G_i; (b) graph $G_i - V_{i-1}^{ex}$ without redundant edges; (c) graph G at $j = 1$; (d) graph G at $j = 2$.

$$\mathrm{sum}\,RP^* = \max_{T_{i,j} \in G} \sum_{o \in T_{i,j}} RP_{o,i} \quad (23)$$

using customized branch and bound procedure.

(7) Allot external vehicles for orders that are not exhausted. Select the minimum value of n_o^{ext} such that

$$n_o^{\mathrm{ext}} \cdot cap_{\mathrm{ext}} + \sum_{i=1}^{nC}\sum_{j=1}^{nc_i} \mathrm{cap}(o,T_{i,j}) \geq Q_o \quad \forall o \in O$$

Terminate.

Illustration. Consider the graph in Figure 4a. Let vertex o_6 be exhausted on some category $C_{i'}$, $i' < i$.

- Step 3: For the first carrier ($j = 1$), the graph G_i is updated by dropping the vertices exhausted on the previous categories C_i', $i' < i$. Removing o_6 and redundant indirect connections like (o_1, o_4), we get the graph G_i in Figure 4b. This graph $G_{i,1} \subseteq G_i$ now is the part of G_i starting from $\hat{o}_{i,1}$.
- Step 5: In the case of $j > 1$, the exhausted vertices are removed (Step 5c). For $j = 2$, $V_1^{\mathrm{ex}} = \{o_3\}$ and this vertex is removed from $G_{i,2} \subseteq G_i$. Now, some of the previously redundant connections may become necessary due to exhausted vertices. On the graph G at $j = 1$ (see Figure 4c,d), the vertex

o_3 is exhausted and according to step 5d, the edges (o_1,o_4) and (o_2,o_5) are re-established.

• Step 6: The maximum order tree is $T_{i,1} = o_1 \rightarrow o_3 \rightarrow o_4$ (see Figure 4c). Let o_3 be exhausted as a result of this selection.

Customized branch and bound

The procedure to find the single branch maximum order tree starting from $\hat{o}_{i,j}$ can be implemented in the form of a simple recursive fathoming algorithm. The algorithm starts from the vertex $\hat{o}_{i,j}$ and at every iteration selects a vertex from the separation space, which is the set of child vertices $Ch(o)$ of the current vertices o. Branching to this selected vertex is done followed by updating of the lower bound. Whether or not further branching should be done is decided based on fathoming criteria. This algorithm has the following two fathoming criteria:

(1) (Bounds) At any vertex o, the upper bound on the objective $sumRP + Ub(o)$ ($sumRP$ is the sum of resource potentials of vertices up to the current vertex o) is compared with the value of the best current solution Lb. This is the typical bounding step in a generic branch and bound algorithm. However, the upper bounds $Ub_{i,j}(o)$ at any vertex can be initialized to that obtained for the previous carrier ($Ub_{i,j-1}(o)$) except for $j = 1$ when it is set to ∞ (similar to a conventional branch and bound). This is possible due to the exploitation of properties of the order graph (see Lemma 1).

(2) (Previous search) If a vertex has already been searched on a previous visit, then it is redundant to search again.

The customized bounding criteria discussed above emphasize the power of contextual optimization that exploits these specific properties of the order graph. Also, determination of the separation space and new solution at any vertex are trivial steps, considering the amiable structure of the order graph for a branch and bound algorithm employing depth first search. (At each vertex o, the separation space is simply the set of child vertices $Ch(o)$.) If the lower bound on the objective value $sumRP + L(o)$ at any vertex o exceeds the current best Lb, then the maximum single branch tree is updated and the algorithm backtracks to the previous vertex.

Illustration. Consider the graph $G_{i,j}$ on carrier $c_{i,j}$ (see Figure 5).

The algorithm starts at $\hat{o}_{i,j}$. Since $\hat{o}_{i,j}$ has not been visited previously, the tree is augmented ($T = \{\hat{o}\}$) and $sumRP$ is incremented ($sumRP = 0$, $RP_{\hat{o}_{i,j,i}} = 0$). This is a branching step. Now, the separation space of $\hat{o}_{i,j}$ is $Ch(\hat{o}_{i,j}) = \{o_1,o_2,o_3\}$ and branching will proceed on these vertices.

• (Fathoming on bounds) Say, $o = o_1$ is selected first. The separation space of o_1 is $Ch(o_1) = \{o_4,o_3\}$. For illustrating the bounding step, let's say o_4 is selected first. Further branching and back-tracking from o_4 will produce the best tree, $T_{i,j}\{\hat{o},o_1,o_4,o_7\}$, when the algorithm back-tracks from o_4 to o_1. The current lower bound is updated to $Lb = 1.5$. Now, when o_3 is selected, $Ub_{i,j}(o_3) = Ub_{i,j-1}(o_3) = 0.8$. Therefore $sumRP + Ub(o_3) = 1.0 < Lb = 1.5$ and, hence, o_3 is fathomed on the basis of bounds.

• (Fathoming on previous search) When $k = 2$, $T = \{\hat{o},o_2\}$ the separation space of o_2 is $Ch(o_2) = \{o_4,o_5\}$. Now, if the choice $o = o_4$ is made then since o_4 has been completely searched previously from o_1, fathoming on the basis of previous search is achieved. Now, the tree $T \cup o \cup T_o^* = \{\hat{o},o_2\} \cup o_4 \cup \{o_7\} = \{\hat{o},o_2,o_4,o_7\}$ is better than the current best $T^* = \{\hat{o},o_1,o_4,o_7\}$ and the current best is updated.

The algorithm will finally terminate when it has branched and backtracked to $\hat{o}_{i,j}$.

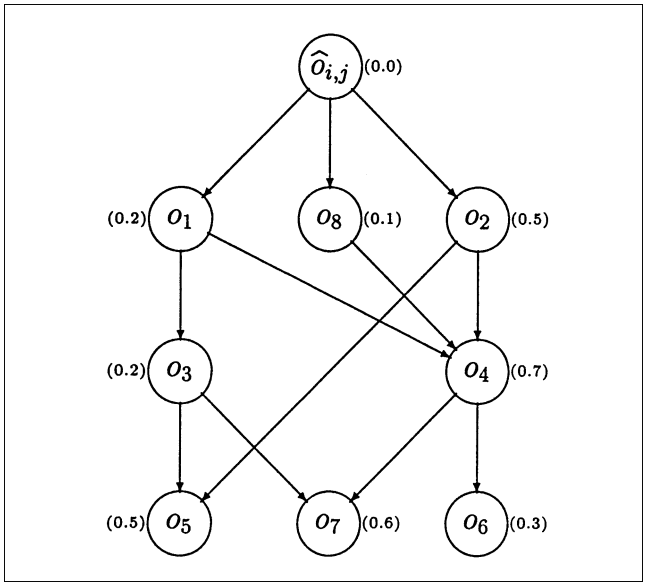

Figure 5. Customized branch and bound.

Artificial connections

If the selection of orders on carriers is done sequentially considering only one carrier at a time, the decisions made on one carrier $c_{i,j}$ may affect the selection opportunities available on subsequent carriers $c_{i,j'} j' > j$. Now this can only happen if the graph G changes from one carrier to another as a consequence of the orders selected already. The connectivity between orders can change if the selected maximum tree contains exhausted vertices (see Definition 4).

Consider again the graph in Figure 3 but with $RP_{o_1,i} = 0.6$. If orders o_1 and o_4 are exhausted with respect to some carrier $c_{i,j}$ after the tree $T_{i,j-1} = \{o_1,o_4,o_6\}$ has been selected on $c_{i,j-1}$, then this breaks the connection (o_2,o_4), and, hence, the choice $\{o_2,o_4,o_6\}$ is out of question. The best choice left is $T_{i,j} = \{o_3,o_5\}$. There is an obviously better solution in

$$T_{i,j-1} = \{o_1,o_3,o_5\} \qquad T_{i,j} = \{o_2,o_4,o_6\} \qquad (24)$$

This counter example shows that an algorithm built on the maximum sum of resource potentials on a tree can yield nonoptimal solutions, although it gets the best solution for a single carrier on a stand-alone basis.

Now, consider a different scenario, which could lead to nonoptimal solutions due to exhausted vertices. Assume that o_1 is exhausted with respect to $c_{i,j}$ and o_2 cannot be allocated on $c_{i,j}$ because $SP_{pp_{i,j},p_{o_2}} = 0$. This leaves out $T_{i,j} = (o_4,o_6)$ as the best choice again missing the optimum since o_1 and o_2 could have swapped positions to obtain the solution in Eq. 24, since o_2 could be allotted on $c_{i,j}$.

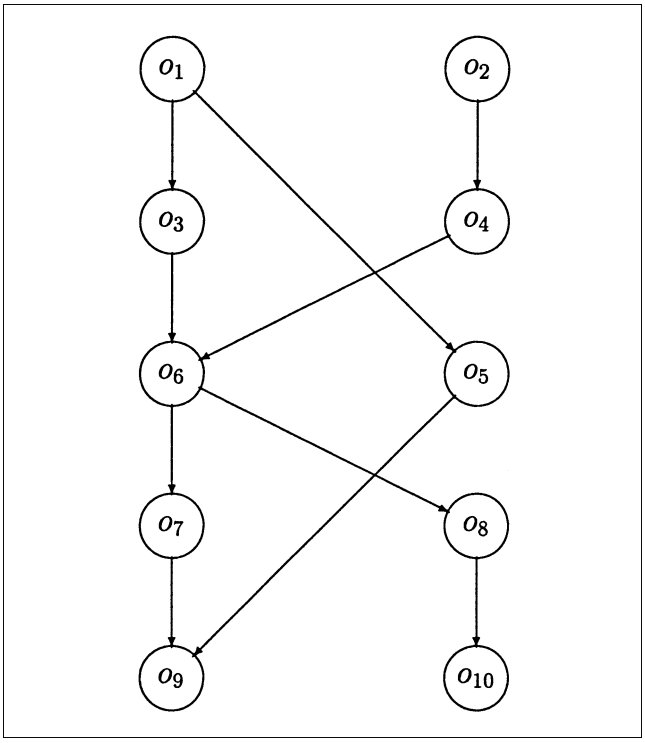

Figure 6. Artificial connections.

In order to account for the nonoptimality caused by exhausted vertices, a concept of adding artificial edge connection is proposed. These artificial edges will account for the potentially lost connections due to the removal of exhausted vertices from the graph. Unlike the real edges, the artificial edges are not just an ordered pair of vertices. Before a generic definition of an artificial edge is put forward, the concept is best illustrated by an example.

Consider the graph of Figure 6. If orders o_1 and o_6 are exhausted due to the selection of $\{o_1,o_3,o_6,o_7,o_9\}$ on $c_{i,j-1}$, then the graph looses the potential connection (o_4,o_6) on $c_{i,j}$. If an artificial edge $e^a=(o_4,o_5)$ and $V^{\text{cut}}(e^a)=o_3$ is introduced, then the choice $\{o_2,o_4,o_5,o_9\}$ will imply

$$T_{i,j}=\{o_2,o_4,o_6,o_7,o_9\}$$

$$T_{i,j-1}=\{o_1,o_5,o_9\}$$

The cut-off vertex (not to be confused with the cut vertex commonly used in graph theory literature) o_3 is removed after the artificial edge is exchanged for real edges. The introduction of artificial edges can therefore help restore the potentially lost connections. In the above example, note that the vertex o_5 acts as a kind of substitute for o_6 on $c_{i,j}$, which can then be transferred to restore the appropriate real connections. Such a vertex is termed as a transferable vertex. The concepts of artificial edge, cut-off vertices and transferable vertices need to be defined formally so that they can be used in generic scenarios in the actual algorithm. The artificial edges should always be replaceable by real edges so as to generate a feasible solution. Since the artificial edge is formed in reference to the trees $T_{i,j}$ already selected, it is essential to keep a record of these and additional connections between them which in turn will facilitate the formation of artificial edges.

Definition 6 (Aggregate Graph, Artificial Edge, Cut-Off vertices, Transferable Vertices). *An aggregate graph* $AG_{i,j}$ is defined as the graph of all single branch trees selected by the MOT algorithm on graphs $G_{i,j'}$ ($j' < j$, $c_{i,j}$ is the current carrier on which MOT is operating) of previous carriers. An artificial edge is a connection between two vertices o^p and o^c where $(o^p,o^c)\notin G$ such that there exists a child $o^{tc}\in AG_{i,j}$ of o^p and there exists a vertex $o^{tp}\in AG_{i,j}$ as part of the tree $o^{tp}\rightarrow T^{\text{cut}}\rightarrow o^{tc}\in AG_{ij}$ such that o^c is the child of o^{tp}. The vertices $V^{\text{cut}}\in T^{\text{cut}}$ are called cut-off vertices. The vertices o^{tp} and o^{tc} are the parent and child transferable vertices, respectively.

Branch and Bound with Artificial Connections. In order to account for the effect of exhausted vertices, the concept of introducing artificial edges can now be utilized into the branch and bound algorithm. The selection of any vertex o from the separation space is done after checking for three criteria.

(1) If o forms a cycle on the selected tree, then branching on o is redundant since a cycle does not make any positive contribution to the resource potential of the tree (see Lemma 2, in the next section).

(2) If a vertex is a cut-off vertex of an artificial edge selected on the current tree, then further branching is redundant because equivalent or better solutions can be obtained through other paths (see Lemma 3).

(3) A vertex forming an artificial edge leads to an infeasible solution if at least one of the corresponding cut-off vertices is either a cut-off vertex or o^{tp} or o^{tc} of another artificial edge already included on the current tree. However, search along such a vertex is redundant due to Lemma 3.

Updating Connections with Cut-off Vertices. Apart from the change in criteria for branching, the increments/decrements in the objective *sumRP* and the lower bound $Lb(o)$ will change to account for the fact that, corresponding to the choice of an artificial edge, there will be a negative weight associated with the removal of the cut-off vertices. Hence, for an artificial edge $e^a=(o,o')$, the updating of the objective and the lower bound on each iteration will be $RP_{o',i}-wCUT(o,o')$ where $wCUT(o,o')$ is the sum of the weight (resource potentials) of the cut-off vertices corresponding to the artificial edge. Also, the set cut-off vertices $V^{\text{cut}}_{i,j}$ should be brought back into the graph G_i. Hence, step 5c of the MOT algorithm should change to the following:

$$5.\ (c)G=\left(G_{i,j}-V^{ex}_{i,j-1}\right)\cup V^{\text{cut}}_{i,j-1}$$

Re-arrangement of Artificial Edges into Feasible Trees. If a maximal tree T^* includes artificial vertices, then the cut-off vertices need to be introduced back in the graph G_i and the trees $T_{i,j}\in AG_{i,j}$ need to be re-arranged so that they represent a feasible solution. The following procedure is used for such a re-arrangement. This procedure should be performed after finding the maximal tree in the MOT algorithm in Step 6. This procedure simply forms the connections (o^p,o^{tc}) and (o^{tp},o^c) for every artificial edge $(o^p,\ o^c)$ and restores a feasible solution from the one containing artificial edges.

Computational Issues

Computational issues related to the MOT algorithm formalized in the previous section will be discussed in terms of memory utilization and computing efficiency.

Memory utilization

Order Graph. The order graph can be implemented as a connectivity matrix between orders

$$OG(o,o') = \begin{cases} 0 & \text{if } (o,o') \notin G, \\ 1 & \text{if } (o,o') \in G, \\ & \quad \text{where } e = \text{index of artificial edge} \\ e+1 & \text{if } (o,o') \equiv e^a \in G. \end{cases} \tag{25}$$

Since the same matrix can be used and updated as the algorithm passes from one carrier to another, a $nO \times nO$ matrix is sufficient to implement the order graph along with the artificial edges.

Aggregate Graph and Artificial Edges. The artificial edges have cut-off vertices and the corresponding o^{tp} and o^{tc}. These can be stored as concatenated vectors, with the pointer to the starting location and number of cut-off vertices stored separately

$$\left\{\ldots, o^{tp}, V_c^{\text{cut}}, o^{tc}, \ldots\right\}$$

The aggregate graph is stored as a $nc^{\max} \times |T|^{\max}$ matrix $\boldsymbol{AG}$ where

$$nc^{\max} = \max_{i \in \{1, \ldots, nC\}} nc_i \tag{26}$$

$$|T|^{\max} = \max_{i \in \{1, \ldots, nC\}} \left(\max_{j \in \{1, \ldots, nc_i\}} |T_{i,j}| \right) \tag{27}$$

Maximum Order Tree. For every order/vertex on the graph, the best tree with the maximum resource potential can be stored by simply storing the pointer to the next vertex it is connected to on the maximal tree. This obviates the need to store the entire maximal tree below a vertex. Also, the bounds associated with each vertex can be stored in vectors $nO + nc_i$ long.

Comparison with MILP. The major component in terms of memory requirement in therefore the $nO \times nO$ order connectivity matrix and the $nc^{\max} \times |T|^{\max}$ matrix $\boldsymbol{AG}$. Hence, nO^2 is a measure of the memory requirement for implementing the MOT algorithm. As opposed to this, a conventional MILP formulation would require memory of the order of $(\sum_{i=1}^{nC} nc_i + nO) \cdot nO^3$ corresponding to the LP matrix apart from the memory overheads for storing information on various branches in the branch and bound procedure. The term $(\sum_{i=1}^{nC} nc_i + nO) \cdot nO$ typically has an order of magnitude of 10^3 for industrial problems. Needless to say, this amounts to significant savings in memory. The use of solvers capable of handling sparse matrix to cope with the memory size would only serve to touch the problem superficially, without evolving a representation that naturally supports the precedent-antecedent relationship between customer orders.

Table 5. Comparison between MILP and MOT: Objective and CPU Time

	Objective		CPU Time	
No.	MOT	MILP	MOT	MILP
1	82.8	82.8	3 s	23 min
2	94.9	94.9	3 s	14 min
3	116.5	116.5	2 s	11 min
4	81.4	81.4	3 s	13 min

Computing efficiency

The maximum order tree on each carrier can be found by an exhaustive search. This was found to be reasonable for short-term delivery scheduling in the industry (see Table 5).

A better alternative would be to use one of the branch and bound algorithms with the relevant fathoming criteria. The branch and bound algorithm involving artificial connections is more rigorous compared to the one without them. However, it is recommended that the former should be used only if the latter deviates significantly from the optimum solution. It can be easily shown that, if the set of cut-off vertices of all artificial connections are restricted to be empty sets, then the branch and bound algorithm (with artificial connections) is *first-order* with respect to the number of edges in the graph. Thus, the overall algorithm would be second-order, first-order with respect to the total number of carriers $\sum_i nc_i$ and also with respect to the number of orders.

In addition to this, the algorithm is further accelerated by the validity of the upper bound on each carrier obtained from computations on the previous carrier.

Lemma 1. *While beginning fathoming on G_j the assignment $Ub_j(o) = Ub_{j-1}(o)$ is valid $\forall o \in G_j$. $Ub_j(o) \le Ub_{j-1}(o) \forall o$, $j = 2, \ldots, nc_i$.*

This lemma permits the use of the upper bounds found on an earlier carrier for a later one within the same category, thereby eliminating redundant searches. On the first carrier in each category, the search is simple since it does not involve artificial edges.

The branch and bound search is kept finite by virtue of the cycle redundancy property.

Lemma 2. *Presence of a cycle CYC does not make any positive contribution to the resource potential of any tree T selected in the fathoming procedure for finding the maximal tree*.

Apart from the finiteness of the search, it also eliminates redundant searches when an already selected vertex is encountered. It also makes the entire idea of artificial edges tractable. If this lemma did not hold, multiple artificial edges would have to be accounted for adding a new dimension of complexity to the problem.

The cut-off and transferable vertices can in principle be used in a current branch and bound search. However the following lemma rules out this possibility.

Lemma 3. *If $o \in V^{\text{cut}}(e^a) \cup o^p \cup o^c e^a \in T$, the current selected tree on the branch and bound search, then branching on o is redundant*.

This property relieves the MOT algorithm from accounting for re-use of cut-off vertices on a current search. Apart from this, it contributes towards optimality of the MOT algorithm by addressing the scenario in which selection of artificial

edges with respect to altered trees (as a result of re-arrangement) is obviated.

The lemmas therefore support the notion of artificial edges, which in turn accounts for incorporating missed opportunities on previous carriers, obviating the need for any iterative search or back-tracking that is typical of more generic search methods. Needless to say, this dramatically affects the computational efficiency even in the case with artificial edges. The algorithm has a theoretical performance of $O(ml)$ (where $m = nc_i$ the number of carriers in each category, and l is the number of edges of the graph), even in the general case of using artificial connections. This theoretical guarantee on the performance is possible due to the following result.

Lemma 4 (Previous search). *Every vertex has to be branched into only once in the customized branch and bound algorithm.*

Optimality of the Algorithm

By maximizing the following objective, the MOT algorithm attempts to use the current category or resource type C_i to the maximum possible thereby minimizing the additional cost of allotting these orders to more expensive resources

$$\sum_{i=1}^{nC} \sum_{j=1}^{nc_i} \sum_{o \in T_{i,j}} RP_{o,i} \tag{28}$$

Hence, it indirectly attempts at minimizing the cost function in Eq. 5.

For each category C_i, the following theorem holds (see Eq. 8).

Theorem 1. *The MOT algorithm gives the maximum value of*

$$\sum_{j=1}^{nc_i} \sum_{o \in T_{i,j}} RP_{o,i} \tag{29}$$

for any category C_i.

Conditions relevant to industrial scenarios that result in near optimal solutions for the objective in Eq. 28 will be discussed next.

The optimality of the MOT algorithm on one carrier category (group of carriers of a particular type) does not in general guarantee optimality over multiple carrier categories. It would, however, be worthwhile to reflect upon a typical industrial scenario to assess the seriousness of the impact of this limitation. The lost opportunity in question stems from the fact that exhausted vertices are removed and, hence, other vertices connected to them may not have any good options left (see Figure 7a and b). o_2 and o_4 are not connected to each other on category C_i. Hence, they have to be selected on separate carriers on C_i. However, if (o_1, o_4) would have been selected on $c_{i-1,j} \in C_{i-1}$, then (o_2, o_3) could also have been selected on a single carrier on C_i. Now, any order not exhausted on one category may in theory stand a chance of being in such a situation on the next category. However, a new carrier category will typically provide such a vertex with alternative connections, from vertices newly added to the graph (see Figure 7b). The vertices o_2 and o_4 now have the new edges (o_2,o_6) and (o_5,o_4), respectively.

Definition 7. *A vertex/order o is new with respect to a category C_i if $o \in G_i$ and $o \notin G'_i \forall i' < i$.*

A new vertex may exist due to any/all of the following reasons of industrial relevance:

(1) Some of the categories $C_{i'}$ may not be suitable for transporting the product corresponding to order o, or this mode of transport may not be available for the customer corresponding to this order.

(2) If $Q_o \le cap_i$, then it is not economical to deliver order o by even higher capacity carriers of type $C_{i'}$, since $cap_{i'} \ge cap_i \forall i' < i$.

It is because of the existence of such new vertices that the potentially lost opportunities are compensated and the algorithm converges with near optimal solutions for the industrial case presented in the next section in Industrial application.

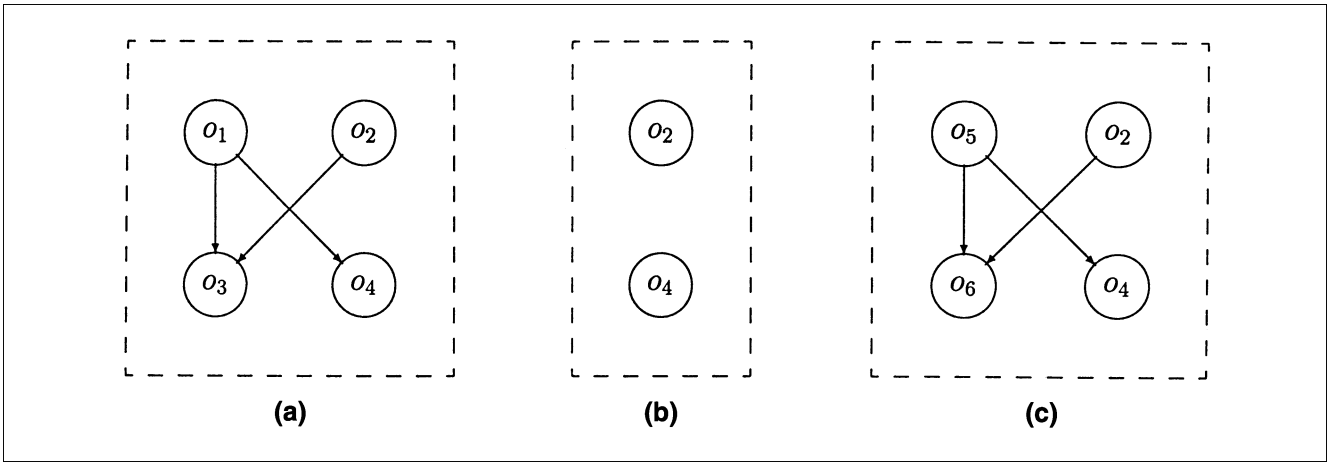

Figure 7. New vertex.

(a) G_{i-1}; (b) $G_i(o_1,o_3 \in V^{ex})$; (c) G_i $(o_5,o_6 \in V^{new})$.

Table 6. Simplified MILP vs. MOT : Objective and CPU Time

No.	Init. Day	No. of Orders	Objective		CPU Time (s)	
			MOT	MILP	MOT	MILP
1	5/6	44	6.16	6.13	5.66	175
2	9/6	55	6.38	6.38	8.73	190
3	10/6	55	7.76	7.74	11.26	350
4	11/6	53	7.89	7.89	9.34	170
5	12/6	54	7.67	7.67	8.62	178
4′	11/6	53	7.89	7.88	equality constraint 10	
5′	12/6	54	7.67	7.67	no penalty on Ext. L	

Evidence of this is found in the next section where an industrial case study is discussed.

Results for an Industrial Case-Study

Problem description

The MOT algorithm was applied to an industrial case study with three carrier categories—trailers, railway wagons, and trucks. These had capacities 20 tons, 10 tons, and 10 tons, respectively. The cost of delivery per unit volume was lowest for trailers and generally highest for trucks with the exception of a few long distance customers. Apart from these carriers, it is possible to hire trucks externally to meet demands that can't be satisfied by any of these. The number of carriers of the three types were $noTr = 8$, $noRW = 58$, and $noL = 12$. The status of these carriers is available in terms of the day on which each of these will return from their previous trip and the last product they delivered. Since the first carrier type had a trailer, two different products possibly to two different customers could be delivered in one trip. However, the customers that could be delivered in one trip were constrained by practical delivery issues related to geographical proximity of customers and whether the trailer had access to the customer site. 26 different grades of a product were to be delivered. The number of demands (orders) varied from 30–200 over a time horizon from 3 to 11 days.

Modeling the problem

The order graph for trailers consisted of vertices that represented order combinations rather than individual orders. The orders to be combined in one trip should have compatible deadlines for delivery. Hence, in this case the set of edges generated for the graph of orders on trailers was governed by E given by

$$E = \{(o,o') | o,o' \in G_i, dd_{o,1} + ret_{o,1} \le dd_{o',1}, SP_{p_o f, p'_o f} \cdot SP_{p_o b, p_{o'} b} = 1\} \quad (30)$$

The external trucks do not have a higher cost compared to the company trucks. Hence, the penalty of using an external carrier is not exactly quantifiable although it is desirable to use company trucks instead of hiring external ones. Appropriate variations of the resource potential for different carrier categories can be engineered to incorporate any penalties that are not quantifiable as in the case of penalties on external carriers discussed above. For trailers and railway wagons, the definition of resource potential is maintained as in Eq. 17. On railway wagons, if there are orders that cannot be delivered by road, then RP is redefined to be $RP_{o,2} = DC_{o,2}$. On trucks, $RP_{o,3} = DC_{o,3}$ which takes care of the penalties on external carriers.

Comparison with MILP

The MOT algorithm was applied with and without artificial connections. Table 6 shows a comparison between the results obtained by applying an MILP formulation and the MOT algorithm without artificial connections for a few small size problems ($nO = 15$, $nt = 3$ days, $noT_r = 16$, $noRW = 25$, $noL = 35$). For problems of larger size, it was not possible to use the MILP approach. Hence, to compare results of problems with more than 15 orders, the problem was simplified, in that the trailers were treated as vehicles of 20 ton capacity that could carry only one product to one customer at a time as against the actual scenario in which two products to two different customers were allowed. With the reduced size ($nt = 3$ days, $noTr = 8$, $noRW = 59$, $noL = 12$), the MILP was solved for orders with a scheduling horizon of three days. The comparison between results obtained from this MILP is shown in Tables 6 and 7. A consistently small deviation ($< 1\%$) from the optimum is observed.

Some interesting limitations of an MILP model were observed. In the case of problem 4 (see Table 6), the MILP solution was found to have a total capacity which was 10 tons more than the required amount. This was because the MILP converged to a solution (within the termination criterion of relative gap = 2%), which delivered 10 tons of an order on a trailer. When the order fulfillment constraint 10 was replaced by an equality constraint, the solution corresponding to 4′ was obtained. However, it may not generally be possible to make

Table 7. Comparison between Simplified MILP and MOT: Distribution Modes

No.	Total Amount (ton)	MOT Solution Deliveries Made By				MILP Solution Deliveries Made By			
		Tr	RW	L	Ext. L	Tr	RW	L	Ext. L
1	700	7	38	18	0	7	33	23	0
2	760	8	41	19	0	7	38	24	0
3	940	12	50	20	0	12	46	24	0
4	930	12	51	18	0	13	48	20	0
5	910	12	45	20	2	12	45	21	1
4′	930	12	51	18	0	12	48	21	0
5′	910	12	45	20	2	12	45	0	22

Table 8. Comparison between Branching with and without Artificial Connections: Objective and CPU Time

No.	Scheduling Horizon (d)	No. of Orders	Objective		CPU Time (s)	
			Without e^a	With e^a	Without e^a	With e^a
1	5	72	0.855420	0.856120	8.4	8.5
2	6	90	1.048272	1.048974	18.5	21.5
3	7	104	1.234074	1.234776	24	37
4	8	127	1.617251	1.617953	45	80
5	9	143	1.823769	1.824471	57	93
6	10	175	1.981830	1.981830	60	110
7	11	195	2.206000	2.206700	79	135

such a replacement if the quantities are fractional amounts of the capacity. For problem 5, the MILP solution was slightly worse than the MOT solution. This was because the use of external trucks has to be penalized in the MILP model to bias the selection of internal trucks. However, in some cases as in problem 5, this may cause the MILP to converge with a higher objective (7.675279) than that of MOT (7.672610). Now, if this penalty is removed, the MILP converges to a lower objective (7.672190, problem 5′) but with a solution that is meaningless (see problem 5′ in Table 7). However, since the selection in the MOT approach is sequential, this does not cause a problem.

Comparison of branching with and without artificial edges

Since the computational time with artificial connection was expected to be higher, an initial Lb on the resource potential was first obtained before branching with artificial connections. This enabled faster convergence due to better use of the fathoming criteria. Table 8 shows a comparison of computational times and objective value for branching with and without artificial connections. Although the objective values for branching with artificial connections are slightly higher than that for branching without them, the former performs better in terms of maximizing the resource potential on the second carrier category of railway wagons. In Table 9 it can be observed that the number of deliveries made using railway wagons is higher for all problems except problem 6 in which case the objective values are equal. However, since the delivery costs for this problem do not necessarily increase monotonically for all orders, this anomalous behavior of getting slightly higher objectives for the branching with artificial connections is found. The increments in computational time in the former branching is found to closely follow the latter.

Table 9. Comparison between Branching With and Without Artificial Connections: Distribution Modes

No.	Total Amount (ton)	Solution—without e^a Deliveries By				Solution—with e^a Deliveries By			
		Tr	RW	L	Ext. L	Tr	RW	L	Ext. L
1	990	13	48	24	1	13	49	23	1
2	1,230	17	59	29	1	17	60	28	1
3	1,360	20	68	27	1	20	69	36	1
4	1,930	28	92	44	1	28	93	43	1
5	2,160	32	104	47	1	32	105	48	1
6	2,370	37	110	52	1	37	110	52	1
7	2,660	43	121	58	1	43	122	57	1

Since it can be easily proven now that branching without artificial connections is a second-order algorithm with respect to the number of orders, it is encouraging to find that the former closely follows this polynomial time relationship, at least in the domain of industrial interest.

Industrial application

The near optimal solutions obtained by the MOT algorithm give evidence of its utility for multiple carrier categories as also of the presence of new vertices on carrier categories 2 and 3. This is expected since not all carriers can be catered by trailers, many demands are only 10 tons, and also not all customers have appropriate access to transportation by trailers and rail. Hence, the presence of new vertices on rail carriers and trucks. The small CPU times enable real-time application for industrial problems.

The performance of the algorithm was also compared with an existing rule-based system used by the company. The new algorithm showed sizeable benefits of about 30 million Yen/year and the schedules could be obtained in a matter of a few seconds. The algorithm used in this case was without artificial connections. Thus, online implementation of the algorithm was possible without any compromise on the quality of the solution.

Extensions to the Framework

Handling changeover time and costs

Changeover time and costs are typically specific to precedent-antecedent pairs of products/orders. Weights can be attached to edges instead of vertices. The resource potential of edge (o,o') will be the same as the resource potential of vertex o' previously. A negative weight equal to the changeover cost can be added to this to account for cleaning costs.

The weights of the new graph G^{ch} can be defined in terms of the original graph G as follows

$$RP_{o,o',i} = RP_{o,i} - C^{ch}(p_o, p'_o) \tag{31}$$

Changeover times for the order pair (o,o') can be used to decide the feasibility of the delivery sequence in addition to the journey/return time for order o. Hence, the definition of edges (see Eq. 14) will change to

$$E = \left\{(o,o') | o,o' \in O, dd_{o'} \geq dd_o + ret_o + t^{ch}_{o,o'}\right\} \tag{32}$$

Scheduling under time windows

The order graph framework presented has been applied to an industrial case study in which vehicles had several (discrete) possible departure times (corresponding to a time table). It is not uncommon, however, to have departure and arrival time windows at the source and destination, respectively. Let T be the length of the relevant time window (at source or destination) and let t be a discrete time instance. An order o can be replaced by orders o_t correspondintg to departure/arrival in time instance t one of which needs to be dispatched. Hence, the number of vertices on the order graph

will be $n \cdot |T|$ where $|T|$ is the number of discrete time intervals in T and the number of edges will be at most $l \cdot |T|^2$. Hence, the performance of the algorithm will be at worst $O(ml \cdot |T|^2)$, although it will actually be proportional to the number of distinct edges.

Thus, the polynomial time performance of the algorithm permits extensions of industrial relevance to the original formulation. These extensions using a general purpose MILP would be really expensive.

Uncertain demand

The demand of any order may be available as a probability distribution $Pr(q,o)$. The probability that the demand is greater than Q is given by the cumulative distribution $Pr(q > Q,o) = \int_Q^\infty Pr(q,o)dq$. The expected cost for each assignment of carrier to order is $DC_{o,i} \cdot Pr(Q_o > Q_o^{\text{assign}},o)$. Hence, the resource potential for uncertain demand can be redefined as

$$RP_{o,i,j} \overset{\text{def}}{=} (UDC_{o,\text{ext}} - UDC_{o,i}) \cdot Pr\left(Q_o > Q_{o,i,j}^{\text{assign}}\right) \quad (33)$$

Notice that the resource potential may reduce from one carrier to another within a carrier category. As carriers are assigned to certain order, the probability of that their demand is left unsatisfied decreases and so do the returns from the associated savings of assigning economical carriers. Orders with lower assignments due to inferior resource potential will then gain priority because of relatively higher probability compared to orders that already have been satisfied to a greater degree. Thus, both cost trade-offs and demand variance aspects are addressed.

Deliveries from multiple locations

Each order may be delivered from a number of possible locations and the carriers may return to different locations with respect to their origin to pick up other orders. This scenario can be easily modeled within the framework by having multiple vertices for each order each representing the site it is delivered from. The edge $(o_s,o'_{s'})$ corresponds to the o delivered from site s and o' from s'. The return corresponding to this sequence will be the journey time from location s to the destination of o and back to the location s'.

Conclusion

A new algorithm was proposed for the optimization of product distribution lines under the philosophy of contextual optimization. A rigorous analysis of various steps of the algorithm shows how problem specific features can be exploited in the form of a graph representation. Sufficient conditions for the optimality of the algorithm reflect industrial scenarios or typical operating conditions. The quality of solutions generated by the MOT algorithm are also evident from the results of an industrial case study.

The graph representation has not only facilitated the development of a tailored algorithm with online implementation, but has also paved the way for developing more generic procedures that can handle a wider class of scheduling problems. Problems involving bulk deliveries from different sites, manpower, and production limitations can now be the next target for exploiting contextual knowledge and optimization.

Notation

$c_{i,j}$ = jth carrier on type C_i
C_i = carrier category of type i
cap^{ext} = capacity of external carrier
cap_i = capacity of any carrier in category C_i
$Ch(o)$ = set of child vertices of order o
CYC = cycle
$da_{i,j}$ = date of arrival of carrier $c_{i,j}$ from trip on previous scheduling horizon
dd_o = departure date for order o
$dd_{o,i}$ = departure date for order o
$dd_{o,s,i}$, $dd_{o,s}$ = departure date for order o from site s
$DC_{i,o}$ = cost of delivering order o on any carrier $c_{i,j} \in C_i$
e = edge (v_1, v_2) of a graph
e^a = artificial edge of a graph
E = set of edges of a graph
$E^d(G)$ = set of direct connections which do not have indirect connections of graph G
$E^{id}(G)$ = set of direct indirect connections of a graph G
G = graph (V,E)
G_i = graph of all orders on category C_i
$G_{i,j}$ = graph of all orders on carrier $c_{i,j}$
$G_{i,j}^o$ = graph of all orders $o' \in G_{i,j}$ such that $(o,o') \in G_{i,j}$
$Lb(o)$, Lb = lower bound on $sumRP(o)$, $Lb = Lb(\hat{o})$
nC = number of internal carrier categories
nc_i = number of carriers in category C_i
nO = number of orders/demands
n_o^{ext} = number of external carriers required to deliver the amount of demand o
nP = number of products
o = customer order
$\hat{o}$ = last order on previous scheduling time horizon
o^c = child vertex
o^{ex} = exhausted order
o^p = parent vertex
o^{tp} = transferable parent vertex
o^{tc} = transferable child vertex
$\hat{o}$ = last order on previous scheduling time horizon
$\hat{o}_{i,j}$ = last order on previous scheduling time horizon on carrier $c_{i,j}$
O = set of all orders
O_i = set of all orders that can be allocated on some carrier category C_i
$pp_{i,j}$ = last product carried by carrier $c_{i,j}$ on previous scheduling horizon
p_o = product corresponding to order/demand o
$Pa(o)$ = set of parent vertices of order o
Q_o = demand volume of order o
$Q_o^{\min}$ = least value $\dot{\iota}$ Q_o that is a linear combination of capacities of various carrier types
$Q_{o,i,j}^{\text{assign}}$ = total volume of order o assigned to carriers of category $C_{i'}$, $i' < i$ and carriers $c_{i,j'}$,j'
$ret_{o,i}$ = return journey time for delivering order o on category C_i
$ret_{o,s,i}$ = return journey time for delivering order o on category C_i from site s
$RP_{o,i}$ = resource potential of order o on category C_i
$\boldsymbol{SP} = SP_{p,p'}$ = product switching matrix
$sumRP$ = sum of resource potentials of vertices (up to the current vertex being branched)
$sumRP^*$ = maximum sum of resource potentials of vertices
T = single branch tree
T^{cut} = tree of cut-off vertices
$T_{i,j}$ = single branch tree corresponding to carrier $c_{i,j}$
T_o = single branch tree starting with vertex o
T_o^* = maximum single branch tree starting with vertex o
TDC = total delivery cost

$TDC^{\min}$ = global minimum of the delivery cost
Ub = upper bound on $sumRP(\hat{o})$
$Ub(o)$ = upper bound on $sumRP(o)$
$Ub_{i,j}(o)$ = upper bound on $sumRP(o)$ on carrier $c_{i,j}$
$UDC_{o,i}$ = unit cost of delivering order o on carrier type C_i vertex
v = vertex
v^{ex} = exhausted vertices
V = set of vertices
V^{cut} = set of cut-off vertices
V^{ex} = set of all exhausted vertices
V_j^{ex} = set of all exhausted vertices on the jth carrier, $c_{i,j}$ for the current category C in the MOT algorithm
$wCUT(o,o')$ = sum of the weight of cut-of vertices corresponding to the artificial edge (o,o')
w.r.t. = with respect to
$y_{i,j,o}$ = binary variable for allocation of carrier $c_{i,j}$ to deliver o

Acknowledgment

We are grateful for the valuable comments of Y. Yokomizo and S. Toshiya and for the financial support from Mitsubishi Chemical Corporation.

Literature Cited

Balas, E., "Disjunctive Programming: Properties of the Convex Hull of Feasible Points," *Discrete Appl. Mathematics*, **89** (1–3), 3 (1998).
Biggs, N. L., *Discrete Mathematics*, Clarendon Press, Oxford (1985).
Brown, G. G., and D. Ronen, "Consolidation of Customer Orders into Truckloads at a Large Manufacturer," *J. of the Operational Res. Society*, **48**(8), 779 (1997).
Carroe, C. C., and J. Tind, "A Cutting Plane Approach to Mixed 0-1 Stochastic Integer Problems," *Eur. J. of Operational Res.*, **101**(2), 306 (1997).
Junginger, W., "On Representatives of Multi-Index Transportation Problems," *Eur. J. of Operational Res.*, **66**, 353 (1993).
Kondili, E., C. C. Pantelides, and R. W. H. Sargent, "A General Algorithm for Short-Term Scheduling of Batch Operations—II. MILP Formulation," *Comput. Chem. Eng.*, **17**(2), 211 (1993).
Laporte, G., "The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms," *Eur. J. of Operational Res.*, **59**, 345 (1992).
Mokashi, S. D., "Contextual Optimisation for Scheduling and Planning of Logistics Systems," PhD Thesis, University of Manchester Institute of Science and Technology, Sackville Street, Manchester, U.K. (Nov. 1999).
Mokashi, S. D., and A. C. Kokossis, "The Maximum Order Tree Algorithm: A New Approach for the Optimal Scheduling of Product Distribution Lines," *Comput. and Chem. Eng.*, S:S1 (1997).
Pinto, J. M., and I. E. Grossmann, "A Logic Based Approach to Scheduling Problems with Resource Constraints," *Comput. and Chem. Eng.*, **21**(8), 801 (1997).
Rajendra Prasad, V., K. P. K. Nair, and Y. P. Aneja, "A Generalized Time-Cost Trade-Off Transporatation Problem," *J. of the Operational Res. Soc.*, **44**(12), 1243 (1993).
Ronen, D., "Allocation of Trips to Trucks Operating from a Single Terminal," *Comput. Ops. Res.*, **19**(5), 445 (1992).
Shah, N., C. C. Pantelides, and R. W. H. Sargent, "A General Algorithm for Short-Term Scheduling of Batch Operations—II. Computational Issues," *Comput. Chem. Eng.*, **17**(2), 229 (1993).
Wilkinson, S. J., N. Shah, and C. C. Pantelides, "Aggregate Modelling of Multipurpose Plant Operation," *Comput. Chem. Eng.*, **19**, S583 (1993).

Manuscript received Feb. 23, 2000, and revision received July 9, 2001.