AN EXACT ALGORITHM FOR THE FULL TRUCKLOAD PICK-UP AND DELIVERY PROBLEM WITH TIME WINDOWS: CONCEPT AND IMPLEMENTATION DETAILS

Gerrit K. Janssens
Kris Braekers
Transportation Research Institute (IMOB) – Hasselt University
Universiteit Hasselt – campus Diepenbeek
Wetenschapspark gebouw 5, 3590 Diepenbeek, Belgium
E-mail: {gerrit.janssens,kris.braekers}@uhasselt.be

KEYWORDS

Exact algorithm, full truckload, pick-up and delivery, set partitioning

ABSTRACT

Intermodal goods transport is characterized by a main transport by rail, barge or seaborne vessel, preceded and followed by short in time but expensive road transport. Many times trucks pick-up or deliver a single container which leads to a full truckload vehicle routing problem in terms of economically efficient transport services. Time windows induced by the customer or due to external situations increase the complexity of an efficient planning by a logistics provider. An exact algorithm for this type of pick-up and delivery problem is developed and the details of its implementation are explained. A set partitioning problem is automatically generated and solved by means of the Lingo software.

1. INTRODUCTION AND LITERATURE REVIEW

Road transport is subject to several problems like congestion, environmental concerns and traffic safety. Intermodal transport is often put forward as a solution to these problems. It can be defined as the combination of at least two modes of transport in a single transport chain, without a change of container for the goods, with most of the route travelled by rail, barge or seaborne vessel and with the shortest possible initial and final journeys by road (Macharis and Bontekoning 2004). An important aspect of intermodal transport is the efficient planning of the pre- and end-haulage activities since these activities constitute a large part of the total costs. Pre- and end-haulage activities involve the transport of containers by road between a container terminal and customer locations. Some containers need to be picked up at a customer location and transported to the terminal, while others are located at the terminal and need to be delivered to a customer location. Time windows may be imposed on these transports. The problem is to find efficient vehicle routes performing all transportation tasks within their time window at minimum cost. Since it is generally assumed that trucks can only transport a single container at a time, routing problems for pre- and end-haulage can be classified as full truckload pick-up and delivery problems (Erera and Smilowitz 2008). When time windows are involved, the problem becomes a Full Truckload Pick-up and Delivery Problem with Time Windows (FT-PDPTW). In this paper, an exact algorithm for solving a FT-PDPTW is presented, which is solved by means of the Lingo software (Schrage, 2007).

A review of the general Pick-up and Delivery Problem (PDP) can be found in Savelsbergh and Sol (1995). For an overview of research on less-than-truckload pick-up and delivery problems, the reader is referred to Parragh et al. (2008a, 2008b). In this section, research on full truckload pick-up and delivery problems is discussed.

Gronalt et al. (2003) present four savings-based heuristics for a FT-PDPTW where full truckloads are transported between distribution centers. A tabu search heuristic for a FT-PDPTW with heterogeneous products and vehicles is proposed by Currie and Salhi (2004). Wang and Regan (2002) study a FT-PDPTW in the context of an intermodal container terminal. A time window partitioning method is used to solve the problem. Another full truckload pick-up and delivery problem in the context of an intermodal container terminal is introduced by Imai et al. (2007). A heuristic based on Lagrangian relaxation is presented. In a first phase, delivery customers are merged with pick-up customers, which may lead to substantial cost and time savings. In a second phase, trucks are assigned to these merged trips. Caris and Janssens (2009) extend the problem of Imai et al. (2007) to a FT-PDPTW by introducing time window constraints at the customer locations. A two-phase insertion heuristic and a local search procedure are proposed. In a subsequent work, a deterministic annealing algorithm is developed to solve the problem (Caris and Janssens 2010).

In Section 2, related literature is reviewed. The problem formulation is presented in Section 3. In Section 4 the proposed exact algorithm is discussed. Finally, a small numerical example is presented in Section 5 and conclusions are drawn in Section 6.

2. SET PARTITIONING APPROACH

Exact algorithms, which guarantee that the best solution found is optimal, are very limited in their applicability due to the NP-complete nature of vehicle routing and scheduling problems. Heuristic methods explore only a small part of the solution space, require less solution time

but do not guarantee that the solution, if found, is a (near) optimal one. A class in between can be referred to as heuristics based on exact methods, as they are often variants of exact methods. The set partitioning approach falls in this class.

The set partitioning approach consists of two phases. The first phase of the approach is a construction phase. Let C be a set with $|C|$ elements and let S_j be subset of C with cost value c_j. In the subset construction phase a number of subsets S_j with cost c_j are generated. The second phase is an optimization problem which selects a subset CS of all subsets S_j : (1) which are mutually disjunct, (2) with a union that is equal to the set C, and (3) with a minimal sum of cost values c_j of the selected subsets in CS.

Algorithms for the first phase are specific for vehicle routing and scheduling problems (full truckload, time windows, …), while algorithms for the second phase are general in nature. The set partitioning approach is a special structured integer programming formulation onto which a vehicle routing problem is mapped. The formulation was first introduced in vehicle routing by Balinski and Quandt (1964). Magnanti (1981) has shown that the vehicle flow formulation of the classical one depot vehicle routing problem may be rewritten into a formulation similar to the set partitioning problem.

In the classical VRP a feasible solution should satisfy a number of constraints like: (1) a vehicle is used at most once; (2) if a vehicle arrives at a customer, it must also depart from there; (3) the load of the vehicle should not exceed its capacity; (4) subtour elimination constraints and (5) integrality of the decision variables. Let FT_v be the set of all feasible solutions for a particular vehicle v with respect to the those constraints and let r be an index to a feasible solution in this set. Such a solution r for a vehicle v is either a feasible route visiting at least one customer, starting and ending at the depot and satisfying the capacity constraint, or an empty route without customers and not departing from the depot.

For each vehicle v, exactly one route of FT_v must be chosen. Let x_r be a 0-1 variable indicating whether route r is chosen or not. Then

$$\sum_{r \in FT_v} x_r = 1 , \forall\, v \in V \quad .$$

Since each customer c has to be visited exactly once, only one of the routes that contains c in all sets FT_v may be chosen. Let $a_{c,r}$ be a 0-1 coefficient that indicates whether customer c is visited in route r or not. Then

$$\sum_{v \in V} \sum_{r \in FT_v} a_{c,r} x_r = 1, \ \forall\, c \in C.$$

To each route r a cost c_r is assigned. Therefore the objective is the minimization of the total cost of the chosen routes and can be written as

$$\sum_{v \in V} \sum_{r \in FT_v} c_r x_r$$

The objective function with both types of contraints exactly forms the set partitioning problem.

In this exact formulation it is assumed that FT_v is the set of all feasible routes for vehicle v. In many situations it is impossible to construct the entire set of feasible routes and only a subset of 'attractive' routes are considered. The construction of this subset is called the route generation phase. It turns the exact algorithm into a heuristic.

In terms of performance, it is worth considering the type of solution algorithm for the set partitioning problem. A general purpose integer programming code can be used, but as it does not exploit the special structure of set partitioning, it can be applied only to rather small problem instances. A number of special algorithms have been developed for solving the set partitioning problem so that larger instances can be solved. Sometimes information from the specific application, viz the VRP, might be used in ordering rows in the tableau to speed up the solution process. An extensive description of the approaches can be found in Balas and Padberg (1976). Later some other algorithms have been developed by Albers (1980), Fisher and Kedia (1986), and Christofides and Paixâo (1993).

The problem under study in this paper is a variant of the classical VRP. Any variant or extension of the classical VRP implies some changes in either the route generation phase or the optimization phase. The variant under study differs from the classical VRP in the following aspects: (1) time windows are imposed; (2) loading is both of the pick-up and the delivery types; and (3) a vehicle can be used multiple times during the planning period.

3. EXACT ALGORITHM

3.1 Algorithm logic

The *full truckload routing problem* is defined on a network $G=(V,A)$ where the customers are located at the nodes (set V). A special node {0} is added and should be interpreted as the single depot. The set V consists of the union of two mutual exclusive sets representing both the 'delivery customers' (set V^D) and the 'pick-up customers' (set V^P), i.e. $V = V^D \cup V^P$ and $V^D \cap V^P = \emptyset$. The members of V^D are designated by d_i (i=1..n_d) and those of V^P by p_j (j=1..n_p).

A *route* is defined as a finite sequence, starting and ending at {0}, of customers. A route is called *feasible* if it satisfies two types of constraints, i.e. logical constraints and temporal constraints. The logical constraints refer to the physical load as a truck is able to carry only one full load (container). A route contains either a single customer or multiple customers.

A *feasible route* can be grammatically described as:

<feasible route> ::= {0} <feasible sequence> {0}

<feasible sequence> ::= <single customer> | <multiple customer sequence>

<single customer> ::= <delivery customer> | <pick-up customer>

<multiple customer sequence> ::= <ordered customer pair> | <single customer> {0} < feasible sequence> | <ordered customer pair> {0} <feasible sequence>

<ordered customer pair> ::= <delivery customer> <pick-up customer>

Let C_k be a set of customers $\{c_1, c_2, \ldots, c_l\}$ of cardinality l. The set C_k may be projected into a finite set of sequences (routes). A procedure needs to be developed to generate the set of feasible sequences related to set C_k. At least one logically feasible sequence can be obtained from the set C_k (i.e. the route consists only of single customer sequences).

The logically feasible sequences, which have been generated, need to satisfy the temporal constraints too. The temporal constraints represent time windows within which the service at the customer's site should start, and a depot time window (opening hours of the depot).
The output of the procedure is the required input for a *set partitioning problem*, which is formulated as follows:

$$min \sum_{i=1}^{s} c_i x_i$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_i = 1 \ (i = 1..s)$$

$$x_i \in \{0,1\}$$

The components of the optimization model should be interpreted as follows:

- *i* represents an index of a customer set, from which at least one logically-and-feasible sequence can be generated;
- c_i represents the minimum cost among the logically-and-feasible sequences generated from set *i*;
- a_{ij} takes value 1 if customer *j* is visited in a sequence generated from set *i* and 0 otherwise;
- x_i is the decision variable which takes value 1 if the minimum-cost sequence from set *i* is included in the dispatching plan and 0 otherwise.

The algorithm solves the full-truckload vehicle routing problem with time windows to the optimal value. The procedure can also be used as a heuristic through reduction of the computational effort. This reduction may be realized in two ways as the procedure consists of two main phases: (1) a preprocessing phase in which all logically-and-temporal feasible routes are generated; and (2) the solution of the set partitioning problem. The procedure in the preprocessing phase works in an iterative way. An iteration *l* includes all operations related to the cardinality of the customer sets equal to *l*, starting from $l = 1$ and increasing it in each iteration by 1. The procedure has a stopping criterion which is described below. By specifying, as a user parameter, a maximal cardinality, the number of generated routes is limited and makes the optimal algorithm a heuristic. Solving the set partitioning problem by means of a heuristic instead of the 0-1 programming optimal algorithm also makes the optimal algorithm a heuristic.
A customer set of size *l* is called *live at size l* if at least one logically-and-temporal feasible sequence can be generated from the set. The iterative procedure stops after step *l'* if no customer sets live at size *l'* can be found. During step *l* of the procedure, customer sets of size *l* are generated by the union of a customer set, live at size *l-1*, and an element (customer) not included in the set of size *l-1*. For this operation only information on the customer sets, live at size *l-1*, is required.
In order to generate logically feasible sequences from a set C_k, the set needs to be partitioned into singletons (single customers) and pairs (ordered customer pairs), and an ordering amongst them. The length of a sequence (number of trips in the route) is indicated by L_{kh} (h=1,…) , with $|C_k|/2 \leq L_{kh} \leq |C_k|$. Let the set of logically feasible sequences, generated from C_k be denoted by $Seq(C_k)$. The elements of the set, which satisfy the temporal constraints, build up the set of logically-and-temporal feasible sequences, denoted by $TSeq(C_k)$, for which holds $TSeq(C_k) \subset Seq(C_k)$.
Let us call the newly generated customer set of size *l* consisting of the *base set* (live at *l-1*) and the *additional customer*. The logical-and-temporal feasible sequences of size *l-1* have been stored. The sequences to be tested for the set, consisting of the union of the base set and the additional customer, are generated by insertion of the additional customer at the head of the sequence, at the tail of the sequence, and al all places within the sequence. Two actions have to be taken: (1) a logically feasible sequence at size *l* needs to be generated; and (2) the temporal constraints have to be tested.
The logically feasible sequences to be generated depend on (1) the type of additional customer and (2) on its neighbour(s) in the existing sequences (or non-existing in case the route is initialized by a first customer). The type of additional customer is either a delivery customer (*dnew)* or a pick-up customer (*pnew*). The customer(s) served immediately before (resp. after) the additional customer is (are) indicated by *pi* or *di* (resp. *pj* or *dj*) in case of single customer and by *(di,pi)* (resp. *(dj,pj)* in case of ordered pairs. Other customers, either before or after the immediate neighbours of the additional customer are not explicitly mentioned in the following rules:

- If the additional customer is the *first customer* to initialize a route

Case 1: delivery customer
{0} *dnew* {0}
Case 2: pick-up customer
{0} *pnew* {0}

- If the additional customer is at the *head of the sequence* (string left of additional customer is empty)

Case 1: delivery customer, followed by a singleton (*remaining* customers to the right are not explicitly indicated, only by …)
{0} *dnew* {0} *dj* {0} …
{0} *dnew* {0} *pj* {0} …
{0} *(dnew,pj)* {0} …
Case 2: delivery customer, followed by an ordered pair
{0} *dnew* {0} *(dj,pj)* {0} …
Case 3: pick-up customer, followed by a singleton
{0} *pnew*{0} *dj* {0} …
{0} *pnew* {0} *pj* {0} …
Case 4: pick-up customer, followed by an ordered pair
{0} *pnew*{0} *(dj,pj)* {0} …

- If the additional customer is at the *tail of the sequence* (string right of additional customer is empty)

Case 1: delivery customer, preceded by a singleton
… {0} *di* {0} *dnew* {0}
… {0} *pi* {0} *dnew* {0}
Case 2: delivery customer, preceded by an ordered pair
… {0} *(di,pi)* {0} *dnew* {0}
Case 3: pick-up customer, preceded by a singleton
… {0} *di* {0} *pnew* {0}
… {0} *(di,pnew)* {0}
… {0} *pi* {0} *pnew* {0}
Case 4: pick-up customer, preceded by an ordered pair
… {0} *(di,pi)* {0} *pnew* {0}

- If the additional customer is *within the sequence* (strings right and left of the additional customer are non-empty)

Case 1: delivery customer, left and right are singletons
… {0} *pi* {0} *dnew* {0} *pj* {0} …
… {0} *pi* {0} *(dnew,pj)* {0} …
… {0} *pi* {0} *dnew* {0} dj {0} …
… {0} *di* {0} *dnew* {0} pj {0} …
… {0} *di* {0} *(dnew,pj)* {0} …
… {0} *di* {0} *dnew* {0} *dj* {0} …
Case 2: delivery customer, left is ordered pair, right is singleton
… {0} *(di,pi)* {0} *dnew* {0} *pj* {0} …
… {0} *(di,pi)* {0} *(dnew,pj)* {0} …
… {0} *(di,pi)* {0} *dnew* {0} *dj* {0} …
Case 3: delivery customer, left is singleton, right is ordered pair
… {0} *pi* {0} *dnew* {0} *(dj,pj)* {0} …
… {0} *di* {0} *dnew* {0} *(dj,pj)* {0} …
Case 4: delivery customer, left and right is ordered pair
… {0} *(di,pi)* {0} *dnew* {0} *(dj,pj)* {0} …
Case 5: pick-up customer, left and right are singletons
… {0} *pi* {0} *pnew* {0} *pj* {0} …
… {0} pi {0} *pnew* {0} *dj* {0} …
… {0} *di* {0} *pnew* {0} *dj* {0} …
… {0} *(di,pnew)* {0} *dj* {0} …
… {0} *di* {0} *pnew* {0} *pj* {0} …
… {0} *(di,pnew)* {0} *pj* {0} …
Case 6: pick-up customer, left is ordered pair, right is singleton
… {0} *(di, pi)* {0} *pnew* {0} *pj* {0} …
… {0} *(di,pi)* {0} *pnew* {0} *dj* {0} …
Case 7: pick-up customer, left is singleton, right is ordered pair
… {0} *di* {0} *pnew* {0} *(dj,pj)* {0} …
… {0} *(di,pnew)* {0} *(dj,pj)* {0} …
… {0} pi {0} *pnew* {0} *(dj,pj)* {0} …
Case 8: pick-up customer, left and right is ordered pair
… {0} *(di,pi)* {0} *pnew* {0} *(dj,pj)* {0} …

3.2 File management

Input data
The input data related to the customers are read from two text files. A first file contains a matrix of distances (expressed as integers) from every delivery point to every pick-up point. A second file contains the distances to the depot both from delivery customers as from pick-up customers, as well as the earliest and latest start-of-service times (time windows).
Input data also come as parameters. These parameters are hard coded, so they are not read from an input file. They include:

the name of the first input file
the name of the second input file
the number of pick-up customers
the number of delivery customers
the service time at the customer's site
the depot closing time.

The first and the second parameters are the names of the text files (including .txt). The third and fourth parameters are required because of the use of static data structures (vectors and matrices). A vector of length equal to the number of delivery (resp. pick-up) customers is used for (1) distances from the customer to the depot, and (2) earliest and latest start-of-service times at the customer's site. A two-dimensional matrix is defined of size 'number of delivery customers' by 'number of pick-up customers' used for the distances between delivery and pick-up customers. The fifth parameter assumes that the service time is equal at all customer's sites. The sixth parameter refers to the time epoch at which any truck should return at the latest (assuming earliest time of departure from the depot is equal to 0).

Work file data
A customer set may have multiple sequences but a sequence belongs to only one customer set. This 1:N relationship might be used in a database programming environment but not in an environment using only sequential (text) files. Therefore the link between, customer sets and sequences must be explicitly coupled in a single data file. The program uses a data file type in which customer sets are defined and in which, after each customer set definition, the logically-and-temporal feasible sequences are defined. A customer set without logically-and-temporal feasible sequences is not included in the file.
As the procedure works in an iterative way, increasing the size of the customer set by 1 in each iteration, two work files of the same type are in use. A first file (called the *old work file*) contains all customers sets and their related sequences at level *l-1* while a second file (called the *new work file*) contains all customers sets and their related sequences at level *l*.

Archive file
A customer set which contains at least one logically-and-temporal feasible sequence is a candidate for inclusion into the set partitioning problem solution. During the iterative process these customer sets are stored. While creating the

work file called the *new work file* it can be checked which of the feasible sequences lead to the lowest cost sequence. The value of this lowest cost is required for the formulation of the set partitioning problem. Also the sequence, related to this minimal cost value, is stored. This is not a requirement for solving the set partitioning problem, but the information is needed to publish the list of customers in a specific route which has been selected in the optimal solution of the problem.

Set_partitioning_problem file
Once the archive file has been completed and the procedure has satisfied its stopping criterion, a program translates the information from the archive file into Lingo-code for the formulation of the 0-1 integer linear programming problem which, as a text file, is ready to be solved by the Lingo software.

The algorithms has a bad worst-case complexity. In case the time windows are hardly restrictive a huge amount of subsets could be eligible as routes, making the set partitioning problem hard to solve. With relatively narrow time windows the complexity is much lower and the problem much easier to handle.

4. NUMERICAL EXAMPLE

In this section, a small numerical example with tree pick-up customers ($V^P = \{p_1, p_2, p_3\}$) and three delivery customers ($V^D = \{d_1, d_2, d_3\}$) is presented. The time windows of the depot (0) and customers are shown in Table 1. Distances between the customers and the depot and between delivery and pick-up customers are shown in Table 2. Customer service time is assumed to be 10.

Table 1: Time windows

Customer	Time window
0	[0, 360]
d_1	[35, 84]
d_2	[253, 278]
d_3	[133, 177]
p_1	[69, 82]
p_2	[242, 252]
p_3	[174, 180]

Table 2: Distances

	0	p_1	p_2	p_3
0	0	63	48	49
d_1	9	57	40	41
d_2	47	42	5	11
d_3	36	45	13	16

Using the procedure presented in the previous section, eighteen feasible routes or sequences can be found. Six of these contain only a single customer, while there are nine sequences of two customers and three sequences of three customers. Solving the set partitioning problem results in the optimal solution which is shown in Table 3. Three vehicles are used and the total distance travelled is 416.

Table 3: Results

Vehicle	Distance	Sequence
1	99	$0 - d_1 - p_3 - 0$
2	220	$0 - d_2 - p_1 - 0$
3	97	$0 - d_3 - p_2 - 0$

5. CONCLUSIONS

The full-truckload pick-up and delivery problem is a relevant problem in an intermodal transport context to make the complete transport chain economically efficient. As most of the pre- and end- haulage in intermodal transport is realized by road transport and is relatively expensive, efficient vehicle routing is an economical benefit. The pick-up and delivery problem is NP-complete making optimal solution very difficult, especially when on top time window constraints are added. In this case researchers turn mostly into heuristics, but it might be also reasonable to investigate how exact algorithms behave computationally. The introduction of time windows probably limits the high computational complexity nature of the problem. In the case that the time windows are relatively hard constraining, an exact algorithm might be solved until optimality in a reasonable computing time. This paper shows how such an implementation can be realized in a computationally efficient way.

REFERENCES

Albers, S. 1980. "Implicit enumeration algorithms for the set partitioning problem." *OR Spektrum* 2, No.1, 23-32.

Balas, E. and M.W. Padberg. 1976. "Set partitioning: a survey." *SIAM Review* 18, No.4, 710-760.

Balinski, M. and R. Quandt. 1964. "On an integer program for a delivery problem." *Operations Research* 12, No.2, 300-304.

Caris, A. and G.K. Janssens. 2009. "A local search heuristic for the pre- and end-haulage of intermodal container terminals." *Computers & Operations Research* 36, No.10, 2763-2772.

Caris, A. and G.K. Janssens. 2010. "A deterministic annealing algorithm for the pre- and end-haulage of intermodal container terminals." *International Journal of Computer Aided Engineering and Technology* 2, No.4, 340-355.

Christofides, N. and J. Paixâo. 1993. "Algorithms for large scale set covering problems." *Annals of Operations Research* 43, No.5, 261-277.

Currie, R.H. and S. Salhi. 2004. "A tabu search heuristic for a full-load, multi-terminal, vehicle scheduling problem with backhauling and time windows." *Journal of Mathematical Modelling and Algorithms* 3, No.3, 225-243.

Erera, A.L. and K. Smilowitz. 2008. "Intermodal drayage routing and scheduling". In *Intelligen Freight Transportation*, P. Ioannou (Ed.). Automation and Control Engineering Series, CRC Press, Boca Raton, FL, 171-188.

Fisher, M.L. and P. Kedia. 1986. "A dual algorithm for large scale set paritioning". Working paper No. 894, Krannert Graduate School of Management, Purdue University, West Lafayette, Ind.

Gronalt, M.; R.F. Hartl; and M. Reimann. 2003. “New savings based algorithms for time constrained pickup and delivery of full truckloads.” *European Journal of Operational Research* 151, No.3, 520-535.

Imai, A.; E. Nishimura; and J. Current. 2007. “A Lagrangian relaxation-based heuristic for the vehicle routing with full container load.” *European Journal of Operational Research* 176, No.1, 87-105.

Macharis, C. and Y.M. Bontekoning. 2004. “Opportunities for OR in intermodal freight transport research: A review.” *European Journal of Operational Research* 153, No.2, 400-416.

Magnanti, T.L. 1981. “Combinatorial optimization and vehicle fleet planning: perspectives and prospects.” *Networks* 11, No.2, 179-213.

Parragh, S.N.; K.F. Doerner; and R.F. Hartl. 2008a. “A survey on pickup and delivery problems. Part I: Transportation between customers and depot.” *Journal für Betriebswirtschaft* 58, No.1, 21-51.

Parragh, S.N.; K.F. Doerner; and R.F. Hartl. 2008a. “A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations.” *Journal für Betriebswirtschaft* 58, No.2, 81-117.

Savelsbergh, M.W.P. and M. Sol. 1995. “The general pickup and delivery problem.” *Transportation Science* 29, No.1, 17-29.

Schrage, L. 2007. *Optimization Modeling with Lingo* (6th ed.), Lindo Systems Inc., Chicago, IL (www.lindo.com).

Wang, X. and A.C. Regan. 2002. “Local truckload pickup and delivery with hard time windows constraints.” *Transportation Research Part B:Methodological* 36, No.2, 97-112.

BIOGRAPHY

Gerrit K. Janssens holds a Ph.D. in Computer Science from the Free University of Brussels (VUB). Currently he is Professor of Operations Management and Logistics at the Hasselt University, Belgium within the Faculty of Business Administration. He also holds the CPIM certificate of the American Production and Inventory Control Society (APICS). During the last eighteen years he has been several times visiting faculty in universities in South-East Asia and in Southern Africa. His main research interests include the development and application of operations research models in production and distribution logistics.

Kris Braekers graduated as Master in Business Economics with a major in Operations Management and Logistics at Hasselt University in 2008. Currently he is preparing a PhD in Applied Economic Sciences at Hasselt University. He is a member of the research group Logistics at the Transportation Research Institute (IMOB) of Hasselt University. His main research interests include modeling empty container management and vehicle routing issues in intermodal freight transport using Operations Research techniques."