View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Ghent University Academic Bibliography

MISTA 2013

# Modeling Real-World Vehicle Routing Problems with Dependencies

**Thijs Walcarius · Maarten Houbraken · Pieter Audenaert · Mario Pickavet · Piet Demeester**

## 1 Introduction

In this research, we consider a novel data model that can be used to solve the Vehicle Routing Problem (VRP) with several of its specializations encountered in the real world.

The basic *Vehicle Routing Problem* is an optimization problem in which a number of customers have to be served with a fleet of vehicles, while minimizing the cost for doing so. As this problem has many applications in the fields of transportation, logistics and distribution, several specializations exist. The *Vehicle Routing Problem with Dependencies* (VRPD) adds restrictions to the order in which customers are served. A dependency between customers indicates a "must happen before"-relationship: $t_{c1} < t_{c2}$. The *Capacitated Vehicle Routing Problem* (CVRP) constrains the maximum amount of cargo that a vehicle can carry at any time. The *Vehicle Routing Problem with Pick-up and Delivery* (VRPPD) is a further specialization of the CVRP and the VRPD, in which cargo is picked up at a customer's location, and has to be delivered to another customer. The delivery event can only happen after the pickup event has been completed. The *Vehicle Routing Problem with Time Windows* (VRPTW) is a variation in which each event has a time window in which it has to be completed: when the vehicle arrives before the earliest allowed time, it has to wait at that location until the time window opens. Ending an event after the time window is prohibited. In the basic VRP, all vehicles start and end at the same location, called the depot. In the *Multiple Depot Verhicle Routing Problem* (MDVRP), this restriction is loosened, and each vehicle can have its own location.

Most current research focuses on modeling the VRP and its extensions as an *Integer Programming* (IP) problem. In this research, we propose a novel approach, in which the problem is modeled in a directed acyclic graph (DAG). Combined with heuristics that incorporate domain knowledge of the VRP and its specializations, it proves to be an efficient approach.

Thijs Walcarius, Maarten Houbraken, Mario Pickavet and Piet Demeester
IBCN, INTEC, Ghent University, Belgium - iMinds
E-mail: `firstname.lastname`@intec.ugent.be

## 2 Modeling the VRP as a DAG

Let $G = (V, A)$ be a directed graph. Each vertex $v$ represents an event, consisting of a location $Loc(v)$ and duration $D(v)$. It can also contain a timestamp $t_v$, that signifies when the event is executed. When the event isn't part of the planning, this timestamp field is left empty. Depending on the type of VRP-problem being solved, additional properties can be added. In our model, there are two types of vertices: *event vertices* and *depot vertices*. An *event vertex* $v$ represents a customer being serviced. In the CVRP it has a capacity delta $\Delta_v$. A *depot vertex* represents the depot where the vehicle leaves from and returns to. As we try to model the problem as an acyclic graph, each depot is duplicated into two *depot vertices*: each tour starts in a *depot start vertex* and ends in a *depot end vertex*.

An edge $e = \langle v_1, v_2 \rangle$ between two vertices denotes a relationship between two events. In our DAG model, we define several types of edges to denote the various relationships between events. A planning edge depicts the most important connection between two events: when a vehicle drives from event $e_1$ to event $e_2$, a planning edge is added from $v_{e1}$ to $v_{e2}$. As this edge signifies a movement of a vehicle, it has a duration $D(e)$: the time it takes to get from $Loc(v_{e1})$ to $Loc(v_{e2})$. In the CVRP, this edge also has a load $l_e$. This load must be between 0 and the maximum capacity of the vehicle. For the VRPD, a second edge type is introduced: the *dependency edge*. This edge signifies that the event $e_1$ must happen before the event $e_2$: $t_{v1} + D(v_1) < t_{v2}$.

For the VRPTW, we enhance the DAG by adding two time windows to each vertex: the *given time window* and the *actual time window*. The *given time window* $[\check{t_v}, \hat{t_v}]$ indicates the earliest start time and the latest end time for the event. For an *event vertex*, this is defined by the customer. In a *depot vertex*, this time window is equal to the operation hours of the vehicle. The *actual time window* $[\check{\tau_v}, \hat{\tau_v}]$ is initially equal to the given time window, but is further constrained by the incoming and outgoing edges. The minimum start time imposed by an edge $e = \langle v_1, v_2 \rangle$ can be computed as follows: $MST(v_2) = MST(v_1) + D(v_1) + D(e)$. Computing the maximum end time can be done analogously. When one of the bounds of the time window changes, this has to be propagated throughout the DAG. Because no cycles are allowed, this updating process cannot result into infinite loops. The window can never become smaller than the duration of the event, ensuring that the planning remains feasible.

### 2.1 Usage of the DAG model

At first, the DAG has to be initialized. For every customer that has to be served, an *event vertex* is created with an empty timestamp. For each vehicle that is available, two *depot vertices* are added. Between these two vertices, a planning edge is added with duration 0.

To add an event $c$ to the route of a vehicle between event $a$ and $b$, the planning edge $\langle v_a, v_b \rangle$ is removed, and two new planning edges $\langle v_a, v_c \rangle$ and $\langle v_c, v_b \rangle$ are added. The durations of these edges are efficiently calculated via a separate routing engine. Additionally, the timestamp of vertex $v_c$ is set, or in the case of the VRPTW, the bounds of the time windows are updated: the maximum end time is recomputed for vertex $a$ and $c$, the minimum start time is recomputed for vertex $b$ and $c$.

### 2.2 Abstracting orders and tasks

In real-world scenarios, logistic companies have to deal with orders that consist of multiple events. Either an order is added to the planning, and all associated events have to be executed, or it is held back and none of the events are executed. When two or more events have to be executed by the same courier (ex. a pickup and delivery of a package), we call them a *task*. One order can consist out of multiple tasks, for example when a package is stored in an intermediate storage and delivered by another courier.

The proposed model only contains events, and makes abstraction of tasks and orders. These are however inherently present in the DAG, as events of the same task typically have a dependency between them. When an order contains multiple tasks, the event of these tasks will also be connected by a dependency edge.

The insertion and removal heuristics used are responsible for inserting and removing all related events to a planning. When events of an order are partially added to a planning, their timing will immediately be reflected in the *actual time windows* of all events that are connected through dependencies. When a partial planning of an order makes the planning infeasible to be executed, this will become clear as the *actual time window* cannot be made smaller than the duration of that event.

## 3 Planning Heuristics using the DAG

Since the formulation of the VRP[1], a tremendous amount of research has been done[2]. We are currently in the process of trading off the most promising heuristics that can be adapted for usage with the DAG described above. Adopting insertion heuristics is currently the most promising approach, as they can benefit from the efficient updating of the *actual time windows*, instead of having to perform extensive checks. Next to that, optimization heuristics like Lin-Kernighan[3] can also benefit from the proposed model, but require multiple operations on the DAG. By employing a styled version of the DAG structure, we focus on efficient implementations of the most occurring primitive operations, trading off execution time with less occurring primitives.

In our approach, we pay attention mostly to the implementation details of the different algorithms and data structures. Careful implementation and corresponding design techniques do pay off dividends, even in the short term. For the current project, further research is needed to determine how this affects the performance of the techniques described above.

**Acknowledgements** This research was conducted with support from IWT, iMinds, Ghent University and a private company.

## References

1. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management science **6**(1), 80–91 (1959)
2. Gendreau, M., Potvin, J.Y., Bräumlaysy, O., Hasle, G., Løkketangen, A.: Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. The vehicle routing problem: Latest advances and new challenges pp. 143–169 (2008)
3. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations research **21**(2), 498–516 (1973)