Vehicle-Routing Problems with Inter-Tour Resource Constraints

Christoph Hempsch [a] Stefan Irnich [a]

[a] *Deutsche Post Endowed Chair of Optimization of Distribution Networks, RWTH Aachen University, Templergraben 64, D-52056 Aachen, Germany.*

Key words: vehicle routing, global and inter-tour resources, efficient local search

1 Introduction

Classical vehicle routing problems (VRPs) are implicitly or explicitly formulated and are solved by considering *resources*: The capacitated VRP (CVRP) has resources for limited vehicle capacities and the VRP with time windows (VRPTW) has resources for service times. In the case of a homogeneous fleet, the limiting resource constraints and resource consumptions are identical for each vehicle. For a heterogeneous fleet, resource constraints and consumptions can differ between specific groups of vehicles. In both cases, the feasibility of a tour depends solely on vehicle-specific resources. Here, we consider constraints for *globally limited resources* that different vehicles compete for. Examples are a restricted number of docking stations at depots, and a limited number of 'long' tours, where long is defined w.r.t. the traveled distance, the number of stops, the arrival time at the depot etc. We devise a general model and solution method and, for the sake of clarity, explain the approach with the example of a VRP with time-varying processing or sorting capacity constraints. Such VRPs arise, for instance, in routing applications for letter mail collection from postboxes or for the pickup of parcels from registered clients: Vehicles collecting mail or parcels arrive at a specific depot over time. The entire volume must be processed (stamped, sorted, labeled with a machine-readable code, commissioned etc.) before a given cut-off time. Moreover, the processing rate at the depot is limited. It may vary over time so that, for each point in time, one can specify a maximum quantity that can be handled in the remaining time interval, i.e., from that point in time until cut-off. While each individual tour may be feasible w.r.t. given time window and vehicle capacity constraints, the feasibility w.r.t. processing capacities is not automatically guaranteed, but requires a staggered arrival of collected mail. Thus, the feasibility of a solution

Email addresses: hempsch@or.rwth-aachen.de (Christoph Hempsch), sirnich@or.rwth-aachen.de (Stefan Irnich).

1 February 2007

depends on the arrival time and collected mail volume of *every single* vehicle at a depot.

The contribution of this chapter is threefold: First, the aim of the new model is to help represent different real-world VRPs with inter-tour constraints in a generic way. The model is mainly based on the unified framework of Irnich (2006b) and utilizes the giant-tour representation (Christofides and Eilon, 1969) and the concept of resource-constrained paths (Desaulniers *et al.*, 1998; Irnich and Desaulniers, 2005). Not only inner-tour but also inter-tour resource constraints are modeled using resource-extension functions (REFs). REFs describe the resource update along a path, i.e., when a vehicle travels from one point to the next. The novelty of this approach is that not only are individual tours considered as resource-feasible paths, but also the entire giant route. By using tailored *reset REFs* connecting the end node of one tour with the start node of the next tour, inner-tour resources are reset, while inter-tour resources are propagated along the entire giant route. This paper clarifies which types of REFs lead to well-structured models, for which the feasibility of a giant route can be efficiently checked.

Second, the new model is intended to support efficient solution procedures that are based on local search (LS). LS-based procedures iteratively build neighbor solutions first and check the feasibility and gain of these afterwards. If straightforwardly implemented, this feasibility check causes an extra effort bounded by the length of a longest tour, which is in general only bounded by $O(Rn)$ for instances of size n with R resources. The methods of (Irnich, 2006b) allow the searching of neighborhoods of size $O(n^k)$ in $O(Rn^k)$ time, thus avoiding an additional factor in the worst-case for cost computations and feasibility checks. We provide sufficient conditions on the REFs that guarantee $O(R)$ feasibility tests for VRPs with inter-tour resource constraints.

Third, the paper presents concepts for applying *sequential search procedures* to inter-tour constrained VRPs in order to further reduce the effort of evaluating a neighborhood of size $O(n^k)$. The goal here is to perform less than $O(Rn^k)$ operations in the average case. Sequential search is a gain-based search-tree pruning method which was first applied to unconstrained problems, such as graph partitioning problems and the symmetric TSP (see Kernighan and Lin, 1970; Christofides and Eilon, 1972). Irnich *et al.* (2006) describe the sequential search approach generically and apply it successfully to the CVRP. Good results have also been obtained for so-called 'rich' VRPs with different kinds of side constraints (see Irnich, 2006b). Here, we show that sequential search enables the fast and efficient solution of *large-scale* multi-depot VRPTW (MDVRPTW) instances with time-varying processing capacities and up to a few hundred collection points. The integration of the LS procedures into a large neighborhood search (LNS) method (Shaw, 1998; Pisinger and Røpke, 2006) leads to an effective metaheuristic, which can easily be adapted to other VRPs with inter-tour constraints.

The chapter is structured as follows: The next section focuses on modeling aspects, starting with models for the MDVRPTW, continuing with the incorpora-

tion of time-varying processing capacity constraints, and ending with the generic inter-tour model and its applications. Section 3 summarizes the techniques used for efficient local search and sketches the implemented LNS metaheuristic. Computational results are presented in Section 4. We show that the proposed modeling and solution approach is helpful to perform new types of studies in which the impact of inter-tour constraints on the structure and cost of solutions is analyzed. Final conclusions are drawn in Section 5.

2 Models for the VRP with Inter-Tour Constraints

The above-mentioned VRP with time-varying processing-capacity constraints serves as an example motivating the giant-tour model and heuristic solution approach. The VRP we are considering is an extension of the MDVRPTW. We start with a non-standard formulation utilizing REFs. This MDVRPTW model has similarities with the unified model of Desaulniers *et al.* (1998). Our goal is to provide a formulation from which we can easily derive a new model. This new model will represent a solution as a single resource-feasible path.

2.1 The Multiple-Depot VRP with Time Windows

The MDVRPTW is defined on a network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ with node set $\mathcal{V}$ and arc set $\mathcal{A}$. As usual, at customer $i \in \mathcal{C} \subset \mathcal{V}$, a quantity of q_i needs to be collected by a single visit of a vehicle. Each customer i allows the start of the service (=collection) within the time window $[e_i, l_i]$.

Let K be the set of vehicles. Since we assume that each vehicle performs exactly one tour during the planning horizon, K is also the set of tours. Each tour $k \in K$ starts at its origin $o(k) \in \mathcal{V}$, ends at its destination $d(k) \in \mathcal{V}$, and visits customers in between. Side-dependencies may restrict vehicle k to visiting only customers $\mathcal{C}^k \subseteq \mathcal{C}$. Hence, the subnetwork $\mathcal{N}^k = (\mathcal{V}^k, \mathcal{A}^k)$ with nodes $\mathcal{V}^k = \mathcal{C}^k \cup \{o(k), d(k)\}$ describes feasible movements of vehicle k in space. For modeling purposes, it is advantageous to formulate the problem with distinct nodes, which results in $\mathcal{O} = \{o(k) : k \in K\}$ and $\mathcal{D} = \{d(k) : k \in K\}$ both having cardinality $|K|$.

The vehicles $k \in K$ are characterized by the following data: The total quantity collected by vehicle k must not exceed the vehicle capacity Q^k. Time windows $[e_{o(k)}, l_{o(k)}]$ and $[e_{d(k)}, l_{d(k)}]$ restrict the start time and end time of tour k. Travel times t_{ij} and costs c_{ij} for $(i,j) \in \mathcal{A}$ are assumed to be vehicle-independent. Note that additional service times at a node i can always be included in t_{ij} without changing the interpretation of the time windows. The model for the MDVRPTW reads as follows:

$$\min \quad \sum_{k \in K} T^{k,cost}_{d(k)} \tag{1a}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{j:(i,j) \in \mathcal{A}^k} x^k_{ij} = 1 \quad \forall i \in \mathcal{C} \tag{1b}$$

$$\sum_{j:(o(k),j)\in\mathcal{A}^k} x^k_{o(k),j} = \sum_{i:(i,d(k))\in\mathcal{A}^k} x^k_{i,d(k)} = 1 \quad \forall k \in K \tag{1c}$$

$$\sum_{j:(i,j)\in\mathcal{A}^k} x^k_{ij} - \sum_{j:(j,i)\in\mathcal{A}^k} x^k_{ji} = 0 \quad \forall k \in K, i \in \mathcal{V}^k \tag{1d}$$

$$x^k_{ij} \in \{0,1\} \quad \forall k \in K, (i,j) \in \mathcal{A}^k \tag{1e}$$

$$x^k_{ij}(T^{k,cost}_i + c_{ij} - T^{k,cost}_j) \leq 0 \quad \forall k \in K, (i,j) \in \mathcal{A}^k \tag{2a}$$

$$T^{k,cost}_i \geq 0 \quad \forall k \in K, i \in \mathcal{V}^k \tag{2b}$$

$$x^k_{ij}(T^{k,load}_i + q_j - T^{k,load}_j) \leq 0 \quad \forall k \in K, (i,j) \in \mathcal{A}^k \tag{2c}$$

$$0 \leq T^{k,load}_i \leq Q^k \quad \forall k \in K, i \in \mathcal{V}^k \tag{2d}$$

$$x^k_{ij}(T^{k,time}_i + t_{ij} - T^{k,time}_j) \leq 0 \quad \forall k \in K, (i,j) \in \mathcal{A}^k \tag{2e}$$

$$e_i \leq T^{k,time}_i \leq l_i \quad \forall k \in K, i \in \mathcal{V}^k \tag{2f}$$

This non-linear mathematical programming formulation of the MDVRPTW contains two types of decision variables: First, flow variables x^k_{ij} for $k \in K$ and $(i,j) \in \mathcal{A}^k$ are equal to 1 if arc (i,j) is used in tour k, and 0 otherwise. Second, resource variables $T^{k,r}_i$ represent the consumption of resource $r \in R$ of tour k at node i. For the MDVRPTW, one has to consider the resources $R = \{cost, load, time\}$.

Constraints (1b) ensure that each customer $i \in \mathcal{C}$ is assigned to exactly one tour $k \in K$. A continuous flow (=movement of vehicle k) between origin $o(k)$ and destination $d(k)$ in $\mathcal{N}^k$ is guaranteed by (1c) and (1d). The non-negative resource variables $T^{k,cost}_i$ record the costs of the (partial) tour starting at $o(k)$ and ending at the respective node $i \in \mathcal{V}^k$. The correct update of the tour costs is ensured by (2a): If vehicle k moves directly from i to j, the partial cost $T^{k,cost}_j$ is at least the cost $T^{k,cost}_i$ plus the cost c_{ij} along the arc (i,j). Note that $T^{k,cost}_i$ can always be set to zero if a node i is not visited by vehicle k. Therefore, the objective (1a) exactly determines the cost of all tours. Operational costs on the arcs can be supplemented by fixed costs on arcs $(o(k),i)$ connecting the origin with a first customer. Also the arc $(o(k),d(k))$ can exist in $\mathcal{A}^k$ to represent the empty tour k.

The remaining limited resources, time and load, are modeled by the resource variables $T^{k,time}_i$ and $T^{k,load}_i$, which are constrained to feasible values by (2d) and (2f). Their update is given by (2c) and (2e). The load update (2c) is managed identically to the cost update (2a). The update of the times by (2e) guarantees together with (2f) that $T^{k,time}_j \geq \max\{e_j, T^{k,time}_i + t_{ij}\}$ holds whenever vehicle k uses arc (i,j). Vehicles arriving before the start of the time window have to wait.

It is obvious that the objective and the capacity constraints can also be formulated in a more 'classical' way, e.g., minimize $\sum_k \sum_{ij} c_{ij} x^k_{ij}$ and $\sum_{ij} q_j x^k_{ij} \leq Q^k$ for all $k \in K$. There also exist straightforward linear reformulations of the time updates (2e) using the well-known big-M technique. The point is, however, that the above formulation is more generic, since it handles all three resources identically: The constraints (2a), (2c), and (2e) can be reformulated with REFs, which is more convenient for a graph-theoretic description of the problem. The formulation with REFs is also essential for the application of efficient LS techniques as presented in

Section 3.

2.2 Formulation of Resource Constraints by Classical REFs

Resource constraints for paths can be modeled by means of *(minimal) resource consumptions* and *resource intervals*. Let R be the number of resources. A vector $T = (T^1, \ldots, T^R)^\top \in \mathbb{R}^R$ is called a *resource vector* and its components *resource variables*. T is said to be *not greater* than S if the inequality $T^i \leq S^i$ holds for all components $i \in \{1, \ldots, R\}$, denoted by $T \leq S$. For two resource vectors a and b, the interval $[a, b]$ is defined as the set $\{T \in \mathbb{R}^R : a \leq T \leq b\}$. Resource intervals, also called *resource windows*, are associated with nodes $i \in V$ and are denoted by $[a_i, b_i]$ with $a_i, b_i \in \mathbb{R}^R$. The changes in the (minimum) resource consumptions along each arc $(i, j) \in \mathcal{A}$ are given by a vector $f_{ij} = (f_{ij}^r)_{r=1}^R$ of *resource extension functions* (REFs). An REF $f_{ij}^r : \mathbb{R}^R \to \mathbb{R}$ depends on a resource vector $T_i \in \mathbb{R}^R$, which corresponds to the resource consumption accumulated along a path $(s, \ldots, i)$ from s to i, i.e., up to the tail node i of arc (i, j). The result $f_{ij}(T_i) \in \mathbb{R}^R$ can be interpreted as a resource consumption accumulated along the path $(s, \ldots, i, j)$. For a comprehensive introduction to resource-constrained paths, we refer to (Irnich and Desaulniers, 2005; Irnich, 2006a).

Let $P = (v_0, v_1, \ldots, v_p)$ be any path in $\mathcal{N}$. Path P is *resource-feasible* if resource vectors $T_i \in [a_{v_i}, b_{v_i}]$ exist for all $i \in \{0, 1, \ldots, p\}$ such that $f_{v_{i-1}, v_i}(T_{i-1}) \leq T_i$ holds for all $i \in \{1, \ldots, p\}$. We can now re-formulate (2a)–(2f) with resource intervals and REFs: Let M be a sufficiently large number. For each node $i \in \mathcal{V}$, let $a_i = (a_i^{cost}, a_i^{load}, a_i^{time})^\top = (0, 0, e_i)^\top$, $b_d = (b_d^{cost}, b_d^{load}, b_d^{time})^\top = (M, Q^k, l_i)^\top$ for $d = d(k) \in \mathcal{D}$ and $b_i = (M, M, l_i)^\top$ for nodes $i \notin \mathcal{D}$. Moreover, let $t_{ij} = (t_{ij}^{cost}, t_{ij}^{load}, t_{ij}^{time}) = (c_{ij}, q_i, t_{ij})$ for all arcs $(i, j) \in \mathcal{A}$ and define the REF f_{ij} by

$$f_{ij}(T) = \max\{a_i, T + t_{ij}\}. \tag{3}$$

Then (2a)–(2f) is equivalent to

$$T_i^k \in [a_i, b_i] \quad \forall k \in K, i \in \mathcal{V}^k \tag{4a}$$
$$x_{ij}^k (f_{ij}(T_i^k) - T_j^k) \leq \mathbf{0} \quad \forall k \in K, (i, j) \in A^k. \tag{4b}$$

These constraints simply state that the paths $P = P(x^k)$, implied by the routing variables x^k, have to form resource-feasible paths.

2.3 Formulation of Time-Varying Processing Capacities

For the MDVRPTW, we exemplify inter-tour constraints by time-varying processing capacities. First, the limited processing capacities are added to the model (1)–(2) (or (1)+(4)) as non-linear constraints. Second, we show that the same constraints can be formulated more easily with additional resources as resource-feasible path constraints, when the definition of REFs is extended to the giant route.

Tours $k \in K$ deliver their collected load to several depots. Therefore, those destination locations $d(k)$ that represent the same physical location must be grouped:

Let $G = \{g(k) : k \in K\}$ be the set of all depots, where $g(k)$ denotes the depot at which tour k ends. Let n_g be the number of tours ending at depot g so that we can index the tours by $h \in \{1, 2, \ldots, n_g\}$. Moreover, vehicle $k(g, h)$ is the hth vehicle ending its tour at depot g and $K(g) = \{k(g, h) : h = 1, \ldots, n_g\}$ is the set of all tours ending at depot g. Recall that $[e_{d(k)}, l_{d(k)}]$ is the time interval in which tours $k \in K(g)$ can deliver to depot $g = g(k)$. Obviously, the amount to be delivered to depot g after the cut-off time $l_{d(k)}$, denoted by $P^g(l_{d(k)})$, is zero. In general, let $P^g(\tau)$ be the *maximum amount of load* that can be *delivered to depot g after time* τ. For the earliest start of service $e_{d(k)}$ at the depot g, $P^g(e_{d(k)})$ is the *overall quantity* that can be processed at g in the given time horizon $[e_{d(k)}, l_{d(k)}]$.

In the following, we assume that processing capacities are discretized and that τ_ℓ, $\ell \in L$ are the points in time at which processing capacities are checked. Figure 1 depicts a typical processing-capacity function and its discretization. (Note that—in principle—we could make the discretization of time dependent on the depots $g \in G$, but we do not want to overload the notation.)

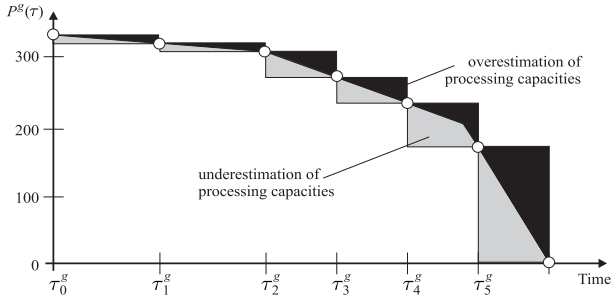

Fig. 1. Example of a Discretization of a Time-Varying Processing Capacity Function

The difficulty in formulating the processing capacity constraints at time τ_ℓ and depot g is that we have to sum up the load of all vehicles $k \in K(g)$, but only if k arrives at g later than time τ_ℓ. In a non-linear formulation, this dependency can be modeled by partial sums over the vehicles $k(g, h)$, $h \in \{1, 2, \ldots, n_g\}$. We get

$$S^g_{\ell,h-1} \leq S^g_{\ell,h} \quad \forall g \in G, \ell \in L, h \in \{2, \ldots, n_g\} \tag{5a}$$

$$(T^{k,time}_{d(k)} - \tau^g_\ell)(S^g_{\ell,h-1} + T^{k,load}_{d(k)} - S^g_{\ell,h}) \leq 0 \quad \forall g \in G, \ell \in L, h \in \{2, \ldots, n_g\}, k = k(g, h) \tag{5b}$$

$$0 \leq S^g_{\ell,h} \leq P^g(\tau_\ell) \quad \forall g \in G, \forall \ell \in L, h \in \{2, \ldots, n_g\} \tag{5c}$$

Herein, $S^g_{\ell,h}$ is the *partial sum* of all loads arriving at depot g later than time τ_ℓ for the first tours $1, 2, \ldots, h$. Inequalities (5a) guarantee that the sequence of partial sums is non-decreasing. The interdependency between the arrival time and collected load of tour k and the corresponding partial sum is modeled by (5b): If tour k arrives late, i.e., $T^{k,time}_{d(k)} > \tau^g_\ell$, then the hth partial sum $S^g_{\ell,h}$ must exceed $S^g_{\ell,h-1}$ by the collected load $T^{k,load}_{d(k)}$. For early arrivals, i.e., $T^{k,time}_{d(k)} \leq \tau^g_\ell$, the constraints (5b) allow $S^g_{\ell,h-1} = S^g_{\ell,h}$. The processing-capacity restrictions are stated by (5c).

The giant-tour representation of a solution is depicted in Figure 2. Here, giant tours are defined as Hamiltonian cycles in the *routing graph* $(\mathcal{V}, \mathcal{A})$, the nodes of which are customer nodes $\mathcal{C}$ as well as start nodes and end nodes of tours, $\mathcal{O}$ and $\mathcal{D}$.

Following the ideas presented in (Irnich, 2006b, p. 4), tour-start and tour-end nodes (o, d) in a feasible route $p = (o, \ldots, d)$ must fulfill a *compatibility relation*. The compatibility relation $\sim$ on $\mathcal{O} \times \mathcal{D}$ introduces vehicle and depot characteristics into the problem. In multi-depot problems, the sets O and D are partitioned according to the $|G|$ depots, e.g., $\mathcal{O} = \mathcal{O}^1 \cup \ldots \cup \mathcal{O}^{|G|}$, $\mathcal{D} = \mathcal{D}^1 \cup \ldots \cup \mathcal{D}^{|G|}$. Pairs $(o, d) \in \mathcal{O}^e \times \mathcal{D}^f$ are compatible $o \sim d$ if and only if $e = f$.

In addition to the arcs of model (1)–(2) (or (1)+(4)), the routing graph also contains *reset arcs* $(d, o) \in \mathcal{D} \times \mathcal{O}$. These reset arcs connect end nodes of one tour with start nodes of another tour. If $(p^1, p^2, \ldots, p^{|K|})$ are the tours forming a feasible solution to the MDVRPTW, the cyclic concatenation of the tours is a giant tour in the routing graph. The corresponding *giant route* is a path (with identical start and end node). It is denoted by $P = P(p^1, p^2, \ldots, p^{|K|})$ and is defined as the concatenation of $p^1, p^2, \ldots, p^{|K|}, o^1$, i.e., of the $|K|$ tours plus the arc connecting the last node $d^{|K|}$ of the last tour $p^{|K|} = (o^{|K|}, \ldots, d^{|K|})$ with the first node o^1 of the first tour $p^1 = (o^1, \ldots, d^1)$.

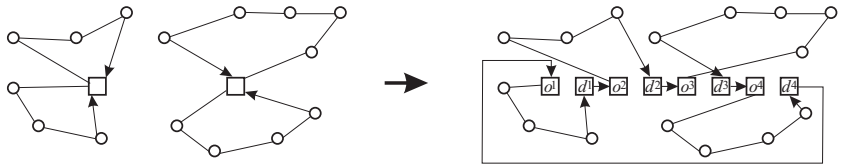

Fig. 2. Giant-Tour Representation

The processing constraints (5) can be equivalently reformulated as resource-feasible path constraints for the giant route P. We define additional resources $r(g, \ell)$ for all pairs $(g, \ell) \in G \times L$. The associated resource variables have the following resource windows at the nodes and REFs on the arcs of the routing graph:

$$T_i^{g,\ell} \in [a_i^{g,\ell}, b_i^{g,\ell}] = [0, P^g(\tau_\ell)] \qquad \forall i \in \mathcal{V} \tag{6a}$$

$$f_{ij}(T)^{g,\ell} = \begin{cases} T^{g,\ell} + T^{load}, & \text{if } T^{time} > \tau_\ell \text{ and } (i,j) = (d(k), o(k')) \text{ reset arc} \\ & \text{and } g = g(k) \\ T^{g,\ell}, & \text{otherwise} \end{cases} \tag{6b}$$

With the definitions $t_{do}^{cost} = 0$ and $t_{do}^{load} = t_{do}^{time} = -M$ for reset arcs $(d, o) \in \mathcal{D} \times \mathcal{O}$, all MDVRPTW resources have well-defined REFs, given by (3) and (6b). (Note that the name *reset arc* refers to the fact that $f_{do}(T)^{load} = 0$ and $f_{do}(T)^{time} = e_o$ holds, i.e., these inner-tour resources are reset to their lower bounds at the tour-start node o.)

2.4 Generic Giant-Tour Model

The generic model for VRPs with inter-tour constraints is the following: Given (a) the routing graph with request/customer nodes, tour-start nodes $\mathcal{O}$ and tour-end nodes $\mathcal{D}$, (b) a compatibility relation $\sim$ between $\mathcal{O}$ and $\mathcal{D}$, and (c) resources, constrained by resource intervals at all nodes, with REFs defined on all (original and reset) arcs of the routing graph, a *giant tour* $(p^1, p^2, \ldots, p^{|K|})$ is *feasible* if its corresponding *giant route* $P = P(p^1, p^2, \ldots, p^{|K|})$ is a resource-feasible path. Recall

that a giant tour has already been defined as a Hamiltonian cycle in the routing graph, the tour-start and tour-end of which respect the compatibility relation $\sim$. The generic *VRP with inter-tour constraints* is the problem of finding a *least-cost feasible giant tour*. The novelty of this definition is that the entire giant route is considered as *one* resource-constrained path (RCP) and that inner-tour as well as inter-tour constraints are all captured in the definition of an RCP by resource intervals and REFs.

The above definition of a VRP with inter-tour constraints has several advantages when heuristic solution methods for solving VRPs are being considered. First, the definition is clear and concise. Second, the concept of RCPs is a very powerful modeling tool, well-known in the context of exact solution approaches in vehicle routing and crew scheduling (Desaulniers *et al.*, 1998). RCPs allow the modeling of many relevant types of constraints for so-called *rich VRPs* including applications with collection and delivery, precedences, side dependencies, multiple use of vehicles, limited waiting time and limited working hours in connection with time windows, time- or load-dependent travel times and costs, complex cost functions and many more aspects (see Irnich, 2006b); additional aspect of modeling with RCPs are coved by (Desaulniers *et al.*, 1998; Avella *et al.*, 2004; Irnich and Desaulniers, 2005; Irnich, 2006a). Third, the definition of the giant tour as a Hamiltonian cycle leads to easier descriptions of local search neighborhoods. For instance, the relocation of a node inside its own tour or into another tour has the same description in the giant-tour representation. The most important advantage is, however, that there are very *efficient* neighborhood search methods available, at least when REFs fulfill some basic requirements. This is the subject of the Section 3.

We now reformulate the MDVRPTW with processing capacities and briefly sketch other inter-tour constraints that can easily be modeled within the same generic framework.

2.4.1 Generic Model for MDVRPTW with Processing Capacities

With definitions (4) and (6) of resource windows and REFs, the MDVRPTW with processing capacities is the problem of finding a least-cost feasible giant route P, where P is resource-feasible w.r.t. resources load, time, and $r(g, \ell)$ for all $(g, \ell) \in G \times L$. If existent, side-dependencies have to be modeled by additional resource constraints (see Irnich, 2006b, p. 19). The consideration of vehicle-dependent capacities Q^k is trivial by defining corresponding resource intervals (4a) at nodes $d(k) \in \mathcal{D}$. However, vehicle dependent costs and travel times can—in principle—be formulated with REFs and additional resources (see Irnich, 2006b, p. 21), but these extensions are not fully compatible with the efficient search methods of Section 3.

2.4.2 Examples of Inter-Tour Resource Constraints

Time-varying processing capacities are a rather complex example of inter-tour resources. Some other simple but practically relevant examples of inter-tour resource constraints are given in the following.

In many real-life, multi-depot problems, the total capacity of the depots is limited. The maximum depot capacity can easily be modeled with the processing constraints introduced in Section 2.3: For vehicle k belonging to depot $g = g(k)$, capacities are only checked at the beginning $e_{d(k)}$ of the processing time window and $P^g(e_{d(k)})$ must be set to the overall quantity that can be processed at depot g.

Also, the number of vehicles being serviced at the same time might be restricted due to a limited number of ramps at the depot. This is, again, a processing capacity, where each tour collects one unit and $P^g(\tau)$ has to be set to the number of available ramps at depot g from time τ until the closing of the depot at time $l_{d(k)}$.

Irnich (2006b, p. 22f) comments on restricting the number of tours with certain characteristics. Examples are a limited number of tours arriving after a certain point in time, traveling more than a given distance or time, collecting more than a certain amount of goods etc. These examples have in common that one resource r_1 is, at tour-end nodes, compared against an upper limit u_1 (non-binding for the individual tours). The number of times this limit is exceeded is recorded by another resource r_2, which is bounded by a upper bound u_2. As long as the resource r_1 is updated by a classical REF of the form (3), one can also limit the number of tours that do not exceed u_1. Hence, it is also possible to restrict the number of tours that arrive early, travel short distances, or collect only a small quantity.

Another interesting task that can be handled by an inter-tour resource is the allocation of a limited vehicle fleet to several depots. Let a fleet of u vehicles be given. In the giant-tour representation, each depot is initially provided with the whole fleet of vehicles, i.e., $u = |\mathcal{O}^1| = \ldots = |\mathcal{O}^{|G|}| = |\mathcal{D}^1| = \ldots = |\mathcal{D}^{|G|}|$. Then, an inter-tour resource globally asserts that the total number of non-empty tours does not exceed u. It is straightforward to extend the model to fleets with multiple vehicle types by using as many inter-tour resources as vehicle types are present.

3 Solution Methods

We have already seen that inter-tour constraints arise naturally in many VRP applications. In particular, the consideration of integrated problems (over multiple depots and extended planning horizons) leads to the concourse of large-scale problem instances with inter-tour constraints. It is, therefore, imperative that heuristic methods should be designed to work both efficiently and effectively.

The solution methodology presented next is based on the *unified framework* of Irnich (2006b), and reference is also made to two earlier papers (Irnich *et al.*, 2006; Irnich, 2006a) for a detailed description of the methods and further questions related to implementation issues.

3.1 Efficient Local Search

Nearly all metaheuristics for VRPs rely on the concept of neighbor solutions, defined by neighborhoods, such as k-Opt and k-Opt* neighborhoods, node relocation and Or-Opt neighborhoods, node and string swap/exchange neighborhoods, and

others (see surveys by Bräysy and Gendreau, 2005a,b; Funke *et al.*, 2005). For all of these neighborhoods, a move from a current solution to a neighbor solution is characterized by the fact that the given giant tour is first split into (a small number of) paths. In the following, these paths are referred to as *segments*. The move permutes the segments - some may be inverted - and they are finally concatenated to form a new giant tour.

A LS algorithm explicitly or implicitly inspects all neighbor solutions and determines the one that is feasible and most improving. There are two aspects of *efficient* LS that we focus on in the following: First, efficient *feasibility tests* are necessary to guarantee that neighborhoods can be explored quickly. It is important to point out here that VRPs with R resource constraints imply an additional factor of at least R in the feasibility tests. Hence, from a *worst case* point of view, the best we can expect are $O(Rn^k)$ time algorithms for searching neighborhoods of size $O(n^k)$. Second, we devise efficient search methods that, in the *average case*, need less than Rn^k steps for fully exploring an $O(n^k)$ neighborhood.

The acceleration of the average case needs further explanation: In the context of node-exchange and edge-exchange neighborhoods, any LS algorithm can be considered a *tree search method*. The tree has depth k for a neighborhood of size $O(n^k)$. In order to accelerate the search, the two main criteria for a reduction of the search space (i.e., pruning the search tree) are *feasibility* and *cost* with two corresponding approaches (Irnich *et al.*, 2006; Irnich, 2006b): *Lexicographic search* is driven by feasibility reductions, i.e., one tries to prove at an early stage $i < k$ that no feasible exchange exists that includes the nodes or edges of the stages $1, \ldots, i$. The concept, as originally introduced by Savelsbergh (1986, 1990), is intrinsically tied to the lexicographic ordering in which neighbor solutions are constructed: In the *innermost loop* of the search algorithm, from one iteration to the next, an inner segment must grow by one node (or a small constant number of nodes), so that so-called *global variables* can be updated in $O(R)$ time. Conversely, *sequential search* is based on the idea of cost-based reductions, i.e., one tries to prove at an early stage $i < k$ that no improvement can be found which includes the nodes or edges of the stages $1, \ldots, i$. It requires, however, that all in-arcs and out-arcs of a node are sorted by increasing cost and moves are decomposable into k cost-independent partial moves (see Irnich *et al.*, 2006). Then, neighbor solutions are generated in such an ordering that partial gains of the partial moves fulfill the *gain criterion* (Lin and Kernighan, 1973; Irnich *et al.*, 2006), i.e., one can restrict the search to those cases where all the partial gains are positive. The idea can be applied in the context of best-improvement as well as first-improvement strategies.

3.1.1 Efficient Feasibility Checks

As presented in the paper by Irnich (2006b), the search procedure can be split into a *preprocessing phase*, in which information for feasibility checks is gathered, and an actual search for the *enumeration* of the neighbor solutions. In the preprocessing phase, generalized REFs are computed for a set of segments. In essence, these *segment REFs* and their *inverses* enable $O(R)$ time feasibility tests. Since any

neighbor solution, represented as a giant tour, results from the concatenation of segments of the current solution, feasibility can be tested by propagating lower and upper bounds of resource consumptions along the segments. Lower bounds have to be propagated by segment REFs, while upper bounds have to be propagated by inverse segment REFs. Although the number of all different segment REFs of a given giant-tour of length n is quadratic, the work of Irnich (2006b) shows that only $O(n^{4/3})$ segment REFs must be a priori computed.

The feasibility test with segment REFs is very similar to the on-the-fly computation of global variables, as suggested in lexicographic search procedures. For instance, time window constraints require the computation of a total travel time, earliest departure time, and a latest arrival time as a global variable of a segment. Kindervater and Savelsbergh (1997) clarify these procedures for 2-opt and Or-opt moves in connection with time windows and precedence constraints as well as for problems with simultaneous deliveries and pickups.

For both methods, lexicographic as well as sequential search, there must hold several assumptions on properties of REFs in order to guarantee $O(R)$ time feasibility tests. All REFs must be computable in $O(R)$ and must be non-decreasing, i.e., $S \leq T$ implies $f_{ij}(S) \leq f_{ij}(T)$. It must be possible to generalize REFs to segments, such that concatenations of segment REFs can be computed and evaluated in $O(R)$ time. Finally, REFs f (of arcs and segments) must be invertible in the sense that $f(T) \leq T'$ is equivalent to $T \leq f^{inv}(T')$ for the inverse REF f^{inv}. These assumptions are—in detail—derived and explained in (Irnich, 2006a).

The assumption about the existence of inverse REFs can be relaxed for some resources r. If a resource r is *non-decreasing* along the *entire* giant-tour and globally constrained by *node-independent* resource windows $[a^r, b^r]$, there is no need to include the resource r in the definition of an inverse REF. The feasibility of a giant route can be directly checked only by the forward propagation of the resource. The overall resource consumption is given at the final node of the newly constructed giant route. (A similar argument was used in (Irnich, 2006a, p. 24) in order to explain that some complex REFs for cost must not necessarily be invertible.) As a consequence, the inter-tour resources $r(g, \ell)$ defined in Section 2.3, do not require an inversion. Hence, $O(R)$ time feasibility checks for VRP with inter-tour resources can be implemented if one can construct and evaluate segment REFs in $O(R)$ time. This important property is shown for the time-varying processing capacity constraints in Section 3.1.3.

3.1.2 Sequential Search

The easiest way to describe the idea of sequential search is by considering the 2-opt* (=crossover) neighborhood, originally suggested by Potvin *et al.* (1989). A 2-opt* move is depicted in Figure 3 and its interpretation is that two different routes in the given giant tour exchange their end-segments. Along the alternating cycle $(t_1, t_2, t_3, t_4, t_1)$ (of deleted and added arcs), the 2-opt* move decomposes into two cost-independent symmetric partial moves, where the first is the deletion of the

arc (t_1, t_2) and insertion of (t_3, t_2), and the second the deletion of the arc (t_3, t_4) and the insertion of (t_1, t_4). For the 2-opt* move to be improving, at least one of the two partial moves has to be improving, i.e., the inserted arc has to be less costly than the removed one (Irnich *et al.*, 2006, p. 2412f). A sequential search algorithm utilizes this property for finding improving moves in the following way: An outer loop determines the node t_1 and the arc (t_1, t_2) to be deleted. The procedure then loops over all in-arcs (t_3, t_2) of t_2 as long as $c_{t_3,t_2} < c_{t_1,t_2}$ holds. All these combinations of t_1, t_2 and t_3 imply that the first partial move is improving. Since t_4 is uniquely determined by t_3, one can also check the overall gain and the feasibility of the 2-opt* move. The case with nodes t_3, t_4, and t_1 is symmetric and, therefore, already covered by the above loops. Note that for restricting the inner loop to cases with $c_{t_3,t_2} < c_{t_1,t_2}$, in-arcs must have been previously sorted by increasing cost and stored in neighbor lists.

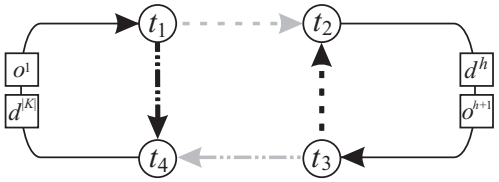

Fig. 3. Principle Sequential Search in the 2-opt* Neighborhood. Partial moves have Gains $g_1 = c_{t_1,t_2} - c_{t_3,t_2}$ and $g_2 = c_{t_3,t_4} - c_{t_1,t_4}$, and $g_1 > 0$ or $g_2 > 0$ must hold for Improving Moves

Sequential search is directly applicable if the cost of a giant tour is the sum of its arcs' costs. Irnich *et al.* (2006) explain decompositions of moves into partial moves for many other types of edge-exchange and node-exchange neighborhoods. Note that the gain criterion can also be generalized to situations where best non-improving moves have to be found.

3.1.3 Resource Extension Functions for Segments

Recall that a segment σ is a sequence of nodes that occur as a sub-path in the giant tour currently under consideration. Figure 3 also visualizes how a 2-opt* move permutes segments. In order to form a neighbor solution, the segment $(o^1, \ldots, t_1)$ is concatenated with the segments $(t_4, \ldots, d^{|K|})$, $(o^{h+1}, \ldots, t_3)$, and $(t_2, \ldots, d^h)$. If segment REFs are given and can be evaluated in $O(R)$ time, the feasibility of the resulting new giant route can also be checked in $O(R)$ time (the number of segments is constant). Thus, we describe next how REFs can be generalized to segments.

For any segment σ, the forward propagation of resources (for given lower bounds T on the resource consumptions) can be computed by a segment REF f_σ of the form

$$f_\sigma(T) = \max\{a_\sigma, T + h_\sigma(T) + t_\sigma\}, \tag{7}$$

where $a_\sigma, t_\sigma \in \mathbb{R}^R$ are resource vectors and $h_\sigma(T)$ is a function $h_\sigma : \mathbb{R}^R \to \mathbb{R}^R$ that takes values $\neq 0$ only for some of the resources $r^{g,\ell}$ and is 0 on all other resources. Moreover, if σ contains *no* reset arc, then $h_\sigma(T) = \mathbf{0}$ for all T, so that (7) is identical to the definition of a classical REF (3). Otherwise, let $(d(k), o(k'))$ be the *first* reset arc in the segment, so that σ can be written as $(\ldots, d(k), o(k'), \ldots)$. Now, we can precisely describe all the coefficients necessary to define h_σ. If a reset

arc exists, let $g_\sigma = g(k)$ be the depot corresponding to the tail node $d(k)$ of the *first* reset arc $(d(k), o(k'))$, and let $\varphi = (\ldots, d(k))$ be the prefix segment of σ up to the first tour-end node $d(k)$. Note that h_σ does not depend on other reset arcs that may be present in σ. If σ contains no reset arc, we define $g_\sigma = \perp$ and φ to be the entire segment σ. Moreover, let $a_\varphi^{time}, t_\varphi^{time}$ and $t_\varphi^{load} \in \mathbb{R}$ be the coefficients that describe the resource consumption for the resources *time* and *load* on the prefix segment φ, i.e., $f_\varphi(T)^{time} = \max\{a_\varphi^{time}, T^{time} + t_\varphi^{time}\}$ is the earliest arrival time at the last node of φ and $f_\varphi(T)^{load} = T^{load} + t_\varphi^{load}$ the collected load. Then,

$$h_\sigma(T)^{g_\sigma,\ell} = \begin{cases} T^{load} + t_\varphi^{load}, \text{ if } \max\{a_\varphi^{time}, T^{time} + t_\varphi^{time}\} > \tau_\ell \\ 0, \qquad\qquad \text{otherwise.} \end{cases}$$

Summing up, the segment REF on segment σ is defined by $(a_\sigma, t_\sigma, g_\sigma, a_\varphi^{time}, t_\varphi^{time}, t_\varphi^{load}) \in \mathbb{R}^R \times \mathbb{R}^R \times (G \cup \{\perp\}) \times \mathbb{R}^3$. Note that also the arc REFs (6b) are of the form (7) with appropriately defined functions $h_\sigma(T)$ having $(a_\varphi^{time}, t_\varphi^{time}, t_\varphi^{load}) = \mathbf{0}$.

What remains to be shown is how one can compute the coefficients of the segment REF of the concatenation of two segments σ_1 and σ_2 in $O(R)$ time. We assume that the last node of σ_1 is identical with the first node of σ_2, and that both segments are described by $(a_{\sigma_1}, t_{\sigma_1}, g_{\sigma_1}, a_{\varphi_1}^{time}, t_{\varphi_1}^{time}, t_{\varphi_1}^{load}), (a_{\sigma_2}, t_{\sigma_2}, g_{\sigma_2}, a_{\varphi_2}^{time}, t_{\varphi_2}^{time}, t_{\varphi_2}^{load}) \in \mathbb{R}^R \times \mathbb{R}^R \times (G \cup \{\perp\}) \times \mathbb{R}^3$. The concatenation $\sigma_1 \oplus \sigma_2$ has a prefix segment denoted by φ (either identical to φ_1 or $\varphi_1 \oplus \varphi_2$ depending on g_{σ_1}) and fulfills

$$\begin{aligned} a_{\sigma_1\oplus\sigma_2} &= f_{\sigma_2}(a_{\sigma_1}) \\ t_{\sigma_1\oplus\sigma_2} &= t_{\sigma_1} + t_{\sigma_2} + h_{\sigma_2}(a_{\sigma_1}) \\ g_{\sigma_1\oplus\sigma_2} &= \begin{cases} g_{\sigma_1}, \text{ if } g_{\sigma_1} \neq \perp \\ g_{\sigma_2}, \text{ otherwise} \end{cases} \\ (a_\varphi^{time}, t_\varphi^{time}, t_\varphi^{load}) &= \begin{cases} (a_{\varphi_1}^{time}, t_{\varphi_1}^{time}, t_{\varphi_1}^{load}), & \text{if } g_{\sigma_1} \neq \perp \\ (\max\{a_{\varphi_2}^{time}, a_{\varphi_1}^{time} + t_{\varphi_2}^{time}\}, t_{\varphi_1}^{time} + t_{\varphi_2}^{time}, t_{\varphi_1}^{load} + t_{\varphi_2}^{load}), & \\ & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

An Example The following example illustrates segment REFs and formula (8). We consider a 2-depot problem with depots $G = \{g, g'\}$, where the processing time window is $[806; 925]$. Processing rates are assumed to be constant with 150 units per hour for depot g and 200 units per hour for depot g'. For the sake of simplicity, the processing capacity functions are discretized at times $\tau_1 = 805$ and $\tau_2 = 835$ only. The resulting capacities are $P^g(\tau_1) = 300$, $P^g(\tau_2) = 225$, $P^{g'}(\tau_1) = 400$ and $P^{g'}(\tau_2) = 300$.

Two segments σ_1 and σ_2 and the associated values for t_{ij}^{time}, $[a_i^{time}, b_i^{time}]$, t_{ij}^{load} (=demand at node j) are given in Figure 4. Both segments contain a reset arc, i.e.,

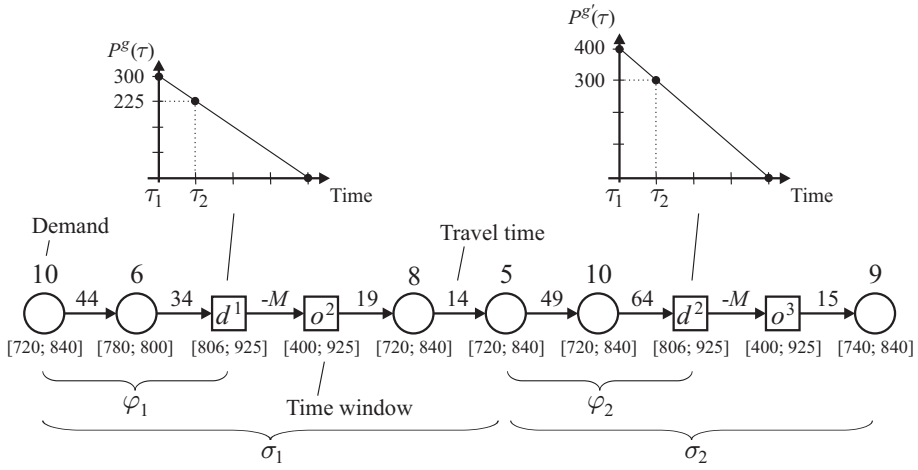

Fig. 4. Two Segments σ_1 and σ_2

(d^1, o^2) for σ_1, and (d^2, o^3) for the segment σ_2. Moreover, tour-end node d^1 belongs to depot g and tour-end node d^2 to depot g'.

From the processing capacity diagrams for both depots, as depicted in Figure 4, it becomes clear that we do not need to check processing capacities at the cut-off time $\tau = 925$, since tours must return to the depots no later than this time. Concluding, the resources to be considered in this example are $\{time, load, (g, \tau_1), (g, \tau_2), (g', \tau_1), (g', \tau_2)\}$ (the computation of costs is trivial and, therefore, left out).

The segment REF f_{σ_1} of the first segment σ_1 is given by

$$f_{\sigma_1}\begin{pmatrix} T^{time} \\ T^{load} \\ T^{g,\tau_1} \\ T^{g,\tau_2} \\ T^{g',\tau_1} \\ T^{g',\tau_2} \end{pmatrix} = \max\left\{ \begin{pmatrix} 734 \\ 13 \\ 6 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} T^{time} \\ T^{load} \\ T^{g,\tau_1} + h_{\sigma_1}(T)^{g,\tau_1} \\ T^{g,\tau_2} + h_{\sigma_1}(T)^{g,\tau_2} \\ T^{g',\tau_1} \\ T^{g',\tau_2} \end{pmatrix} + \begin{pmatrix} -M \\ -M \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\},$$

where $h_{\sigma_1}(T)^{g,\tau_1} = T^{load} + 6$, and $h_{\sigma_1}(T)^{g,\tau_2} = 0$ if $\max\{814, T^{time} + 78\} \leq \tau_2 = 835$ and $h_{\sigma_1}(T)^{g,\tau_2} = T^{load} + 6$, otherwise. The interpretation is simple: The earliest arrival time at d^1 is $814 > \tau_1 = 805$, and, hence, the resource (g, τ_1) is always increased by $T^{load} + 6$, which is the load in the tour arriving at d^1. In general, the arrival time at d^1 is given by $\max\{814, T^{time} + 78\}$, which explains $h_{\sigma_1}(T)^{g,\tau_2}$. Along the entire segment σ_1, the coefficients of f_{σ_1} reflect that the earliest arrival time at the last node of σ_1 is 734 with 13 units of load collected.

The segment REF f_{σ_2} is

$$f_{\sigma_2}\begin{pmatrix} T^{time} \\ T^{load} \\ T^{g,\tau_1} \\ T^{g,\tau_2} \\ T^{g',\tau_1} \\ T^{g',\tau_2} \end{pmatrix} = \max\left\{ \begin{pmatrix} 740 \\ 9 \\ 0 \\ 0 \\ 10 \\ 0 \end{pmatrix}, \begin{pmatrix} T^{time} \\ T^{load} \\ T^{g,\tau_1} \\ T^{g,\tau_2} \\ T^{g',\tau_1} + h_{\sigma_2}(T)^{g',\tau_1} \\ T^{g',\tau_2} + h_{\sigma_2}(T)^{g',\tau_2} \end{pmatrix} + \begin{pmatrix} -M \\ -M \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\},$$

where $h_{\sigma_2}(T)^{g',\tau_1} = T^{load} + 10$, and $h_{\sigma_2}(T)^{g',\tau_2} = 0$ or $h_{\sigma_2}(T)^{g',\tau_2} = T^{load} + 10$ depending on whether $\max\{833, T^{time} + 113\} \leq \tau_2 = 835$ holds or not. Whenever one arrives at the first node of σ_2 three or more minutes later than the earliest

service time (720), the arrival at d^2 is later than τ_2 and resource (g', τ_2) is increased by $T^{load} + 10$.

Using formula (8), the segment REF for the concatenation $\sigma = \sigma_1 \oplus \sigma_2$ is given by

$$f_\sigma \begin{pmatrix} T^{time} \\ T^{load} \\ T^{g,\tau_1} \\ T^{g,\tau_2} \\ T^{g',\tau_1} \\ T^{g',\tau_2} \end{pmatrix} = \max \left\{ \begin{pmatrix} 740 \\ 9 \\ 6 \\ 0 \\ 23 \\ 23 \end{pmatrix}, \begin{pmatrix} T^{time} \\ T^{load} \\ T^{g,\tau_1} + h_\sigma(T)^{g,\tau_1} \\ T^{g,\tau_2} + h_\sigma(T)^{g,\tau_2} \\ T^{g',\tau_1} \\ T^{g',\tau_2} \end{pmatrix} + \begin{pmatrix} -M \\ -M \\ 0 \\ 0 \\ 23 \\ 23 \end{pmatrix} \right\},$$

where $h_\sigma(T) = h_{\sigma_1}(T)$. The interpretation of this result is the following: When traversing the entire segment $\sigma_1 \oplus \sigma_2$, the resulting arrival time and load at the last node is independent from the initial resource consumption T. The tour starting at node o^3 arrives at the last node of σ at time 740 with 9 units of load on board. The concatenation of σ_1 and σ_2 fully determines what happens at depot d^2. 23 units of load arrive after time τ_2 and, hence, T^{g',τ_1} and T^{g',τ_2} are always increased by 23 (which is also the minimum resource consumption). In contrast, processing capacity resources for depot g depend on the collected load and start time at the beginning of the segment σ: Six units of load arrive at d^1 later than τ_1 and, depending on the start time at the first node, possibly also later than τ_2. This is controlled by $h_\sigma(T)^{g,\tau}$, in which the arrival time at the first depot is computed by $\max\{814, T^{time} + 78\}$ and the collected load by $T^{load} + 6$.

3.2 Large Neighborhood Search

Metaheuristics are substantial for producing high-quality solutions because they allow an escape from local minima. This section briefly describes the metaheuristic implemented here, which is obviously only one out of many possible choices for using the efficient LS procedures as a component of a metaheuristic.

To find a first local minimum, various neighborhoods are combined in a VND component (Mladenović and Hansen, 1997; Hansen and Mladenović, 2001). In order to escape from this joint local optimum, a *kick* step is performed. The kick consists of a randomized removal of a subset of nodes that are consecutively reinserted into the giant route. The solution is then re-optimized with the VND component, the resulting solution is compared against the previous local optimum, and accepted with the Metropolis acceptance criterion of simulated annealing (Aarts and Korst, 1989). Hence, the chosen approach has similarities with the large-step Markov chain approach (Martin *et al.*, 1992) and the large neighborhood search (LNS) approach originally proposed by (Shaw, 1998). The difference to the large-step Markov chains is, however, that local optimal solutions of the VND component are used instead of local minima of a single neighborhood. The difference to standard LNS procedures is the use of the Metropolis acceptance criterion. Røpke and Pisinger (2006) also use LNS with the Metropolis acceptance criterion, but their LNS solutions are not re-optimized by LS at all.

There are plenty of choices for defining node removal and node insertion opera-

tors. Over the tested operators (pure random, based on node attributes such as time window length, demand, detour length etc.), the operator that performs best randomly selects 20 'close' customers according to a randomized distance-based selection procedure. Insertion of removed customers is done by building dummy routes containing these customers and by applying the above VND component directly to the resulting giant tour (see also Irnich, 2006b, p. 21). Røpke and Pisinger (2006) choose from among different removal and insertion operators according to scores that are updated by a learning mechanism based on the search history. This may be a beneficial extension to the current implementation.

4 Experimentation

The computations presented in this section aim at two different aspects: First, we show that the solution methodology introduced in (Irnich, 2006b), i.e., giant-tour representation and $O(R)$ feasibility checks by considering the giant-route as a resource-constrained path, lead to highly *efficient* local search-based metaheuristics. Second, we exemplify the usefulness of inter-tour constraints by presenting *new types of studies* that can easily be performed with the methods at hand.

4.1 Efficient Local Search

In order to analyze the efficiency of the proposed LS techniques, we generated a set of 80 MDVRPTW test instances with 100, 200, 400 and 800 nodes (each class with 20 instances). Each instance has between two and five depots. Customers are spread around the depots (according to a normal distribution) such that the service areas of the depots partially overlap. The width of the customer time windows is varied in each group of instances. This creates five groups of MDVRPTW instances. Moreover, four different processing time windows for the depots are chosen for each MDVRPTW instance. The four different processing time windows reflect different situations where processing capacities are more or less binding (from loosely to hardly constrained). Overall, this generates 320 instances of the MDVRPTW with time-varying processing capacities.

Each of 320 instances is solved with the LNS metaheuristic of Section 3.2. VND first alternates between 2-opt, 2-opt*, node swap and node relocation neighborhoods until a joint local optimum is reached. The search procedures for finding improving string-exchange and Or-opt moves (with and without inversion of the relocated segment) are then applied to these local optima. Each VND step ends in a joint local optimum of all seven neighborhoods. 250 kick moves are performed to diversify the search.

Using a similar setup[1] as in (Irnich, 2006b), the absolute performance of the *sequential search* approach is summarized in Table 1: The overall computation time

[1] All algorithms were coded in C++, were compiled in release mode using MS-Visual C++ .NET 2003 version 7.1, and all runs were performed on a standard PC (Intel x86 family 15 model 2 stepping 5, 2.8 GHz, 1GB main memory, on MS-Win 2000).

(second column) to perform the 250 kicks and VND steps does not exceed 15 minutes, even for the largest instances with 800 nodes. The third column shows how often sequential search algorithms were invoked in VND and kick steps. This number does not raise proportional to the size of the instances or size of the neighborhoods, but grows sub-linear. For all instances, the ratio of searches that find an improving neighbor to the total number of searches is stable and between 60% and 68%. We have also computed (fourth column) the average time necessary to perform a single sequential search (including both search phases, segment REF computation and actual tree search). These numbers show that the sequential search procedures are notably fast, in particular for large-scale instances.

Size # Nodes	Avg. Time 250 VND+kick	Avg. Number of Search Steps Performed	Avg. Time per Search
100	35.5 s	12738	2.8 ms
200	119.4 s	17040	7.0 ms
400	279.6 s	19227	14.4 ms
800	716.8 s	22641	31.4 ms

Table 1
Characteristics of the LNS Metaheuristic based on Sequential Search

Finally, we compare the overall computation times of the LNS metaheuristic when either sequential search or lexicographic search procedures are used. Figure 5 depicts the speedup gained by using sequential instead of lexicographic search (the speedup factor is the quotient of the running times). For each size of instances, the five subclasses correspond to increasing widths of the customer time windows. One can clearly see that sequential search outperforms lexicographic search, since the latter takes (on average) between 1.5 and 11.4 times longer. As already observed in (Irnich *et al.*, 2006; Irnich, 2006b), sequential search is more effective for loosely constrained problems and when the size of the instances increases. The impact of increasing customer time windows is that tours get longer and, therefore, instances are less constrained and can be solved significantly faster.

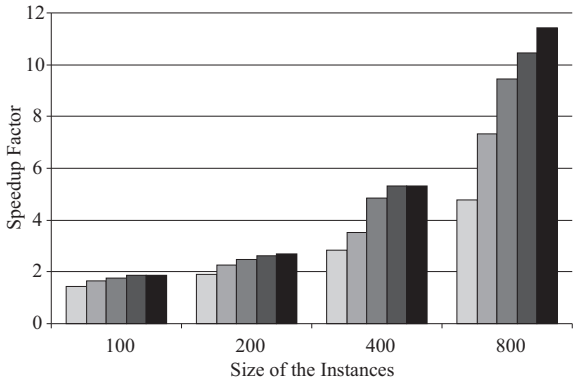

Fig. 5. Acceleration Gained when Sequential Search is used instead of Lexicographic Search

It is worth mentioning that we have also analyzed the four groups of instances with less and less binding processing time windows. For these, the impact on the speedup is less significant and varies by less than 6% within each of the four groups.

4.2 New Types of Studies based on Inter-Tour Resource Constraints

When depots and processing facilities are being planned, the interdependency between transport processes and stationary processes is often disregarded due to the complexity of integrated facility design/layout and transport planning problems. For instance, the dimensioning of the depots as well as the duration of time windows used for processing are unclear. The models and solution techniques presented in this chapter allow such decisions to be studied in an integrated way, at least if it is possible to formulate the stationary processes with inter-tour resource constraints.

4.2.1 Variation of the Cut-Off Times

Several aspects have an impact on the temporal feasibility of solutions: Travel times and service time windows at customers specify the feasibility of the individual tours. The processing rates (i.e., the slopes of $P^g(\tau)$), the length of the processing time windows, and the cut-off times together determine the temporal interdependency between the tours (see also Figure 1). The variation of each of these parameters has consequences for the cost and structure of the resulting VRP solution.

Here we analyze the impact of the cut-off times $l_{d(k)}$ on the cost, the number of tours, and the number of customers that cannot be serviced. We present results for a 4-depot instance with 100 nodes. 30 runs of the LNS metaheuristic are performed, where the cut-off times of all depots are changed from run to run by five minutes. The shape of the processing capacity functions $P^g(\tau)$ is not altered.

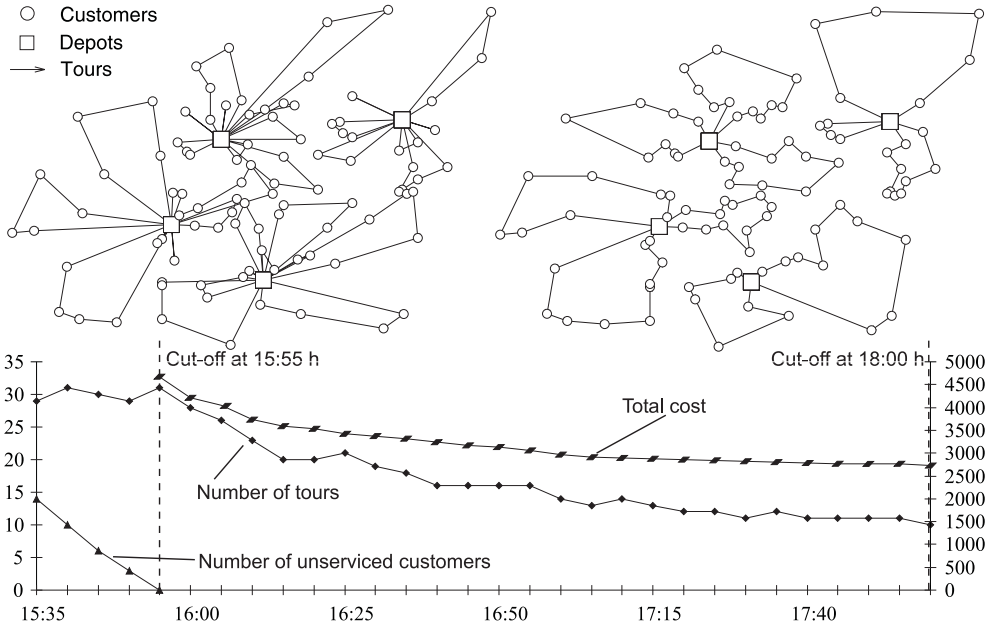

Fig. 6. Simultaneous Variation of Cut-Off Times of all Depots

The diagram at the bottom of Figure 6 shows the transportation costs, the number of tours in the solution, and the number of customers that are not serviced due to the early cut-off times. The later the cut-off times, the less tours must be operated to collect the customers' supply. At the same time, the costs of the solutions de-

crease. Note that we do not use any fixed costs per tour (with fixed costs the effect would be even more drastic). In addition to the cost diagram, the top part of Figure 6 shows the two extremal solutions corresponding to the cut-off times 15:55 h and 18:00 h. In the left tour plan, processing capacities are strongly binding. The result is a relatively high number of tours with only a few customers in each tour. In contrast, with the late cut-off at 18:00 h, tours are not all constrained by the processing capacities.

4.2.2 Optimal Dimensioning of Processing Facilities

Another interesting issue is the determination of the ratio of the processing capacities at different facilities and the impact of processing capacities to transportation (fleet size, cost etc.). For the sake of simplicity, we assume a 2-depot problem,

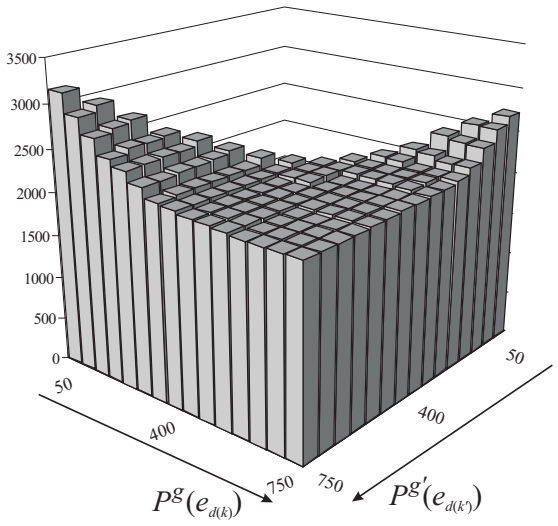

Fig. 7. Variation of Processing Capacities for Two Depots g and g'

where the dimensioning of the machines for both depots is unclear. In order to find an optimal dimensioning of the machines, one can solve several VRPs with inter-tour constraints, where the processing capacities at the two depots g and g' are varied. Figure 7 shows a diagram, in which the resulting transportation cost for each scenario of processing capacities is given. Constant processing rates and fixed processing time windows at g_1 and g_2 are assumed. The processing capacity is then quantified by the pair $(P^g(e_k), P^{g'}(e_{k'}))$ (with $k \in K(g), k' \in K(g')$), cf. Section 2.3). In Figure 7, missing bars correspond to scenarios that are infeasible, since processing capacities are too small to process the entire quantity present at the customers (customers remain unserviced in the VRP solutions). Moreover, each scenario allows an estimation of the transportation costs, which can then be compared with costs in stationary processes (investments for machines, wages for workers etc.). Such a comparison of scenarios means integrated planning of transportation and facility dimensioning.

5 Conclusions

This chapter has focused on the heuristic solution of large-scale VRPs with inter-tour constraints. Inter-tour constraints are those constraints for which the feasibility of a solution depends on properties of several tours and cannot be decided by considering the individual tours separately. Examples are sorting processes at depots that require a staggered arrival of tours, limited number of ramps at depots, and depots with globally limited capacities. Many more examples can be found when transportation and other logistics processes are considered together.

The presented modeling and solution approach can cope with such interdependencies and is based on the unified framework of Irnich (2006b): A solution is represented as a giant tour, i.e., as a single Hamiltonian cycle in the problem-specific routing graph. This representation is advantageous from a modeling point of view, since complex inter-tour constraints can be taken into account by the powerful concept of resource-feasible paths. It has been shown that inter-tour constraints, which are sometimes complicated to formulate in mixed integer programming models, can be easily translated into simple resource-feasible path constraints on the giant route.

The proposed solution method is based on local search (LS), which is one of the most important techniques for improving VRP solutions. It is used as a component in metaheuristics, such as tabu search, GRASP, VND and VNS, or as a postprocessing improvement method in all types of metaheuristics. By considering a giant route as a single resource-feasible path, the unified framework performs LS for many types of VRPs with inter-tour constraints and for all classical LS neighborhoods as efficiently as it does for standard VRPs. The key technique used here is an $O(R)$ time feasibility check for neighbor solutions, where R is the number of resources. The efficiency results from the decomposition of LS procedures into two phases, where the first phase computes segment resource extension functions in $O(Rn^{4/3})$ time. These are used to guarantee $O(R)$ feasibility tests in the second phase, which is the actual search for improving neighbors. Overall, the search takes $O(Rn^k)$ time for node-exchange and edge-exchange neighborhoods of size $O(n^k)$. As a result, different tree search methods, such as lexicographic search and sequential search, are applicable and also allow an acceleration of the search also in the average case.

In the unified framework, model and solution method *both* utilize the giant-tour representation. This is important, since classical *local* search techniques (in particular those using inner-tour neighborhoods) have a quite restricted *local* view of the solution space. In contrast, the LS methods used here can better cope with complicated *global* interdependencies and work, at the same time, highly efficiently. Concluding, the new approach proposed in this chapter shows that large-scale instances of VRPs with inter-tour constraints can be solved efficiently using LS components. It is possible to perform new types of studies, where complex interdependencies between tours and also the impact of other external parameters on structure and costs of VRP solutions can be analyzed. This is much needed for a

more realistic planning of transportation processes in integrated logistics networks.

References

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines.* Wiley, Chichester.

Avella, P., Boccia, M., and Sforza, A. (2004). Resource constrained shortest path problems in path planning for fleet management. *Journal of Mathematical Modelling and Algorithms*, **3**(1), 1–17.

Bräysy, O. and Gendreau, M. (2005a). Vehicle routing with time windows, Part I: Route construction and local search algorithms. *Transportation Science*, **39**(1), 104–118.

Bräysy, O. and Gendreau, M. (2005b). Vehicle routing with time windows, Part II: Metaheuristics. *Transportation Science*, **39**(1), 119–139.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, **20**(3), 309–318.

Christofides, N. and Eilon, S. (1972). Algorithms for large-scale travelling salesman problems. *Operational Research Quarterly*, **23**(4), 511–518.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57–93. Kluwer Academic Publisher, Boston, Dordrecht, London.

Funke, B., Grünert, T., and Irnich, S. (2005). Local search for vehicle routing and scheduling problems: Review and conceptual integration. *Journal of Heuristics*, **11**(4), 267–306.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, **130**(1), 449–467.

Irnich, S. (2006a). Resource extension functions: Properties, inversion, and generalization to segments. Technical Report 2006-01, Deutsche Post Endowed Chair of Optimization of Distribution Networks, RWTH Aachen University, Aachen, Germany. Available at `www.dpor.rwth-aachen.de`, (accepted with minor modifications for publication in *OR Spectrum*).

Irnich, S. (2006b). A unified modeling and solution framework for vehicle routing and local search-based metaheuristics. Technical Report 2006-02, Deutsche Post Endowed Chair of Optimization of Distribution Networks, RWTH Aachen University, Aachen, Germany. Available at `www.dpor.rwth-aachen.de`, (accepted with minor modifications for publication in *INFORMS Journal on Computing*).

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer.

Irnich, S., Funke, B., and Grünert, T. (2006). Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, **33**(8), 2405–2429.

Kernighan, B. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, **49**, 291–307.

Kindervater, G. and Savelsbergh, M. (1997). Vehicle routing: Handling edge exchanges. In E. Aarts and J. Lenstra, editors, *Local Search in Combinatorial Optimization*, chapter 10, pages 337–360. Wiley, Chichester.

Lin, S. and Kernighan, B. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, **21**, 498–516.

Martin, O., Otto, S., and Felten, E. (1992). Large-step Markov chains for the TSP incorporating local search heuristics. *Operations Research Letters*, **11**(4), 219–224.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, **24**, 1097–1100.

Pisinger, D. and Røpke, S. (2006). A general heuristic for vehicle routing problems. *Computers & Operations Research*, **online available**.

Potvin, J., Lapalme, G., and Rousseau, J. (1989). A generalized k-opt exchange procedure for the MTSP. *Information Systems and Operations Research*, **27**(4), 474–481.

Røpke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, **40**(4), 455–472.

Savelsbergh, M. (1986). Local search for routing problems with time windows. In C. Monma, editor, *Algorithms and Software for Optimization, Part I*, volume 4, pages 285–305. Baltzer, Basel.

Savelsbergh, M. (1990). An efficient implementation of local search algorithms for constrained routing problems. *European Journal of Operational Research*, **47**, 75–85.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science*, **1520**, 417–431.