Synchronization in Vehicle Routing— A Survey of VRPs with Multiple Synchronization Constraints

Technical Report LM-2011-02

Michael Drexl
Chair of Logistics Management, Gutenberg School of Management and Economics,
Johannes Gutenberg University Mainz
and
Fraunhofer Centre for Applied Research on Supply Chain Services SCS, Nuremberg

31st January 2011

Abstract

This paper presents a survey of vehicle routing problems with multiple synchronization constraints. These problems exhibit, in addition to the usual task covering constraints, further synchronization requirements between the vehicles, concerning spacial, temporal, and load aspects. They constitute an emerging field in vehicle routing research and are becoming a 'hot' topic. The contribution of the paper is threefold: (i) It presents a classification of different types of synchronization. (ii) It discusses the central issues related to the exact and heuristic solution of such problems. (iii) It comprehensively reviews pertinent literature with respect to applications as well as successful solution approaches, and it identifies promising algorithmic avenues.
Keywords: Survey; Vehicle Routing; Synchronization; Coordination; Transshipment; Trailer

1 Introduction

Vehicle routing problems (VRPs) constitute one of the great success stories of operational research. They have been the subject of intensive study for more than half a century now. This has led to the publication of thousands of scientific papers and to the foundation of more than a hundred software companies worldwide selling commercial vehicle routing software. This development is certainly due to the intellectual challenge VRPs pose as well as to their practical relevance in logistics and transport. Research on VRPs is incessantly ongoing, stimulated by unsolved theoretical problems and continuous input from logistics practice. One generic class of VRP that is receiving more and more interest is denoted here *vehicle routing problems with multiple synchronization constraints (VRPMSs)*: In classical vehicle routing problems, synchronization is necessary between the vehicles with respect to which vehicle visits which customer. VRPMSs are VRPs which exhibit additional synchronization requirements with regard to spacial, temporal, and load aspects. For the purposes of this survey, the following definition applies:

A VRPMS is a vehicle routing problem where more than one vehicle may or must be used to fulfil a task.

It will become clear what is meant by this in a moment: In the next section, an example of a particular VRPMS will make the definition concrete.
VRPMSs constitute an emerging field in VRP research and are becoming a 'hot' topic. This is reflected by the fact that most of the literature surveyed in this paper was published not more than two years ago, and this is a justification for having written the present survey.

The contribution of the paper is threefold: (i) It presents a classification of VRPMS. (ii) It discusses the central issues related to the exact and heuristic solution of VRPMSs. (iii) It analyzes scientific publications on VRPMSs with respect to applications as well as successful solution approaches, and it identifies promising algorithmic avenues.
The presentation of the material assumes familiarity with vehicle routing problems (capacitated VRP, VRP with time windows, pickup-and-delivery problem with time windows, capacitated arc routing problem, dial-a-ride problem etc.) and with the standard modelling and exact and heuristic solution methodologies (mixed-integer programming, branch/cut/price, local/neighbourhood search, metaheuristics). If this is not the case, the reader is referred to Toth/Vigo [121], Golden et al. [64], Desaulniers et al. [48], Funke et al. [58], Ropke [109], Glover/Kochenberger [63].
The rest of the paper is structured as follows. The next section gives a concrete example of an archetypal VRPMS and exemplifies the different types of synchronization identified in this survey. In Section 3, a classification of synchronization (henceforth abbreviated by 's.') is given. Section 4 points out the difficulties concerning the formal modelling and the solution of VRPMSs. Section 5, which forms the main part of this paper, surveys relevant publications by type of s. with respect to applications, models, and algorithms. Section 6 summarizes the central findings of the literature review. Finally, in Section 7, related fields which may offer fruitful input for further study of VRPMSs are identified, and promising directions for future research are proposed.

2 A concrete example: The vehicle routing problem with trailers and transshipments

The vehicle routing problem with trailers and transshipments (VRPTT) was chosen as a concrete example of a VRPMS, because it contains all types of synchronization relevant in this paper. The VRPTT as presented here is a simplified version of the underlying real-world problem. A description of the complete problem can be found in [52]. The research on the VRPTT was motivated by the problem of raw milk collection in Southern Bavaria, Germany: The milk is collected from farmers and is transported to a dairy plant (the *depot*) every day by a heterogeneous fleet of vehicles stationed at the depot, see Figure 1.

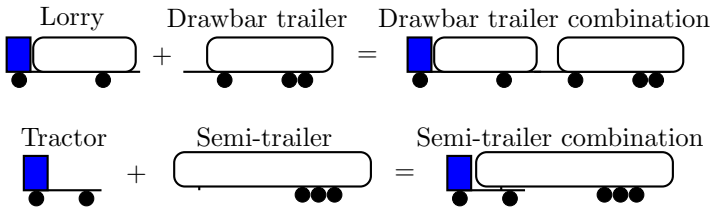

Figure 1: VRPTT fleet

The vehicles differ with respect to two orthogonal criteria: First, lorries and tractors are *autonomous vehicles* able to move in time and space on their own, whereas drawbar trailers and semi-trailers are *non-autonomous vehicles*, which can move in time on their own, but must be pulled by a compatible autonomous vehicle to move in space. Second, lorries and drawbar trailers are *task vehicles* technically equipped to visit customers and collect supply, whereas tractors and semi-trailers are not; they can only be used as *support vehicles*, that is, as mobile depots to which the task vehicles can transfer load. The load transfers can be carried out at *transshipment locations* (TLs) such as parking places.
Most farmers can only be visited by a lorry without a trailer (a *single* lorry) and are hence called *lorry customers*. The other farmers can be visited by a lorry with or without a trailer and are called *trailer customers*. There may be time windows at the customers as well as at the TLs.
All vehicles start and end their routes at the depot. There is no fixed assignment of a trailer to a lorry or of a semi-trailer to a tractor. Any non-autonomous vehicle may be pulled, on the whole

or on a part of its itinerary, by any compatible autonomous vehicle. What is more, any vehicle may transfer its load partially or completely to any other vehicle at any TL arbitrarily often. For technical reasons, at any TL, only one transshipment can be performed at a time, and during any transshipment, only one *active* vehicle can transfer load to one *passive* vehicle. Moreover, the time a transshipment takes depends on the amount of load transferred. An example route plan, which, for simplicity, does not contain support vehicles, is depicted in Figure 2.

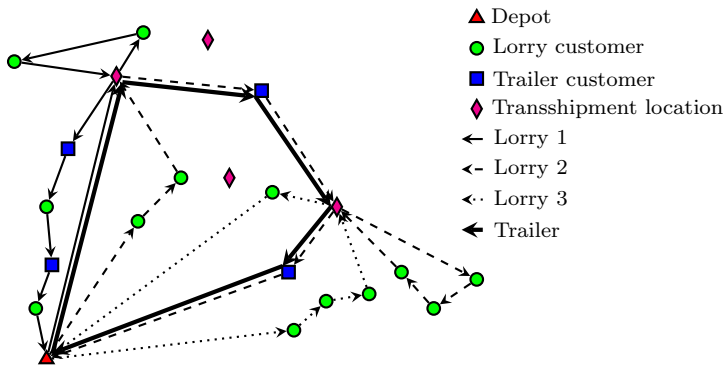

Figure 2: VRPTT example route plan

In the example, lorry 1, together with the trailer, starts at the depot, goes to a TL, decouples the trailer there, visits two lorry customers, returns to the trailer, transfers some load, leaves the trailer there and returns to the depot via two lorry and two trailer customers. Lorry 2 starts at the depot, visits two lorry customers, couples the trailer (after lorry 1 has performed its load transfer), visits a trailer customer, decouples the trailer at another TL, possibly performs a load transfer, visits some lorry customers, returns to the trailer, re-couples it and pulls it back to the depot via a trailer customer. Meanwhile, lorry 3 also starts at the depot, visits some lorry customers, transfers some load to the trailer while lorry 2 is visiting the three lorry customers bottom right, and returns to the depot via another lorry customer. The two TLs in the centre of the figure are not used.
Tractors pull semi-trailers from the depot to TLs, where they either decouple the semi-trailers and return later (in the meantime pulling other semi-trailers) or wait until a semi-trailer has received enough load from other vehicles to be pulled back to the depot. Note that tractors have a capacity of zero and cannot visit any customers; nevertheless, they are useful.
All vehicles may return to the depot for unloading and start new routes arbitrarily often. Vehicles need not carry any load when returning to the depot. Lorries (tractors) need not bring back a drawbar trailer (semi-trailer), neither one they may have pulled when leaving the depot, nor any other.
The problem is to devise routings of minimal total costs for all vehicles (some of which may not be needed), such that the complete supply of all customers is collected and delivered to the depot.
The VRPTT has the following properties in connexion with synchronization:

(i) Customers may be visited by two vehicles.

(ii) Trailers are non-autonomous vehicles that must be pulled by autonomous vehicles to move in space.

(iii) Trailers may be pulled by different autonomous vehicles on their itinerary.

(iv) Support vehicles cannot visit any customers.

(v) Transshipments are possible between arbitrary vehicles.

(vi) At TLs, only one transshipment can be performed at a time.

A transshipment is defined by:

(i) The *location* where the transshipment takes place

(ii) The *point in time* when the transshipment begins

(iii) The *active vehicle*, which transfers all or part of its load

(iv) The *passive vehicle*, which receives load

(v) The *amount of load* transferred

Hence, the central question in the VRPTT is:

Which vehicle transfers how much load when where into which other vehicle?

The main difficulty of the problem lies in the fact that several vehicles may or must participate in fulfilling a task, that is, in collecting a customer's supply and transporting it to the depot. This leads to a close interdependency between the vehicles. This is not usually the case in vehicle routing problems.

3 A classification of synchronization

In standard VRPs, vehicles are independent of one another: A change in one route does not affect any other route. In VRPMSs, by contrast, a change in one route may have effects on other routes; in the worst case, a change in one route may render all other routes infeasible. This is called the *interdependence problem*. In Figure 2, if lorry 1 does not visit the leftmost TL but goes directly to the uppermost lorry customer, the trailer cannot move to the TL, and the other two lorries have no opportunity to transfer load, which may violate their capacity constraints. Addressing the interdependence problem may require different types of synchronization. The following types are identified in this paper:

(i) *Task synchronization*
The fundamental types of object in VRPs are *tasks* and *vehicles*. A task is a mandatory duty, something which must be done and requires zero or more units of some capacity. A vehicle is an autonomous or non-autonomous mobile object which provides zero or more capacity units and can be used to fulfil tasks. Tasks may consist in collecting supply at or delivering demand to one location, in picking up load at one location and delivering this load to another location, in visiting a location to render a service, in selecting and executing a visiting pattern in periodic VRPs etc. Task s. refers to the fact that it must be decided which vehicle(s) fulfil each task. Task s. is what differentiates VRPs from single-vehicle routing problems such as travelling salesman or postman problems. In the VRPTT, all customers must be visited exactly once by exactly one task lorry and by at most one task trailer. The fundamental problem of task s. can thus be stated as follows:

Each task must be performed exactly once by one or more suitable vehicle(s).

(ii) *Operation synchronization*
An *operation* is something that may or must be performed by a vehicle at a location or vertex. Operation s. is the s. of operations of *different vehicles* at the *same or different locations* (or vertices) with regard to the *time* at which the vehicles perform their respective operation at the respective location(s). Consequently, operation s. also decides on the temporal aspects of tasks. Operation s. may induce *dynamic time windows*. A dynamic time window for execution of an operation depends on the execution of another operation. The computation of a schedule for one vehicle without considering schedules for other

vehicles is *not* operation s. In the VRPTT, transshipments are possible only if both the active and the passive vehicle are present at the respective location during the time needed for the load transfer. The dynamic time window for when the passive vehicle involved in a transshipment can start executing its operation, that is, start to receive load, depends on the arrival time of the active vehicle and vice versa, and this arrival time is not given in advance, but is determined in the course of the algorithm. The fundamental problem of operation s. can hence be formulated as follows:

The offset, that is, the time that may elapse between the start of execution of a specified operation by a suitable vehicle at a certain vertex and the start of execution of another specified operation by another suitable vehicle at another certain vertex, must lie within a specified finite interval of zero or positive length, both vehicles must be compatible, and the vertices may be the same one or different ones.

With respect to the consideration of the temporal aspect, three types of operation s. can be distinguished, where the offset is denoted by Δ and the interval within which it must lie by $[a, b]$, $a \leqq b$:

(a) Pure spacial operation s. ($a < b$, $b - a = T$ = length of overall planning horizon $< \infty$ without loss of generality)
This is the case when the temporal aspect of operation s. is ignored and only the spacial aspect is taken into account. Doing so is possible when neither static nor dynamic time windows are considered. Temporal operation s. can then be performed after all routes have been computed, either by an algorithm, by planners who manually assign route start and rendezvous times, or by the drivers themselves.

(b) Operation s. with precedences ($a < b$, $b - a < T$)
This is the case when two vehicles must start executing their respective operation at their respective vertex with a variable offset ($a \leqq \Delta \leqq b$).

(c) Exact operation s. ($a = b$)
This means that two vehicles must either start executing their respective operation at their respective vertex at the same time (simultaneous operation s., $0 = a = \Delta = b$), or with a fixed offset of positive cardinality (deferred operation s., $0 \neq a = \Delta = b$).

(iii) *Movement synchronization*
This refers to the fact that *non-autonomous vehicles* require *autonomous vehicles* to move in space or that non-autonomous vehicles have to join to become able to move in space. In both cases, the movements of at least two *elementary vehicles* must be synchronized with respect to space and time, yielding an autonomous *composite vehicle*. In the VRPTT, drawbar trailers must be pulled by compatible lorries and semi-trailers must be pulled by compatible tractors. The fundamental problem of movement s. can therefore be formulated as follows:

For a vehicle to be able to move along a certain arc, a different but compatible vehicle must move along the same arc at the same time, that is, both vehicles must leave the tail of the arc at the same time, traverse the arc together and reach the tail of the arc at the same time.

This is a restrictive definition and excludes deferred execution of tasks along the same arc. There are two subtypes of movement s.:

(a) Movement s. at the depot
This is the case when two vehicles may join and separate only at the depot, before the start and after the end of a route.

(b) Movement s. en route
This is obviously the case when two vehicles may join and separate different locations which they may visit during their route.

(iv) *Load synchronization*
The amount of capacity used on vehicles when fulfilling a task or performing an operation must be correctly taken into account. In other words, it must always be ensured that the right amount of load is collected or delivered or transshipped. In the VRPTT, for each transshipment, it must be decided how much load is to be transferred. The load the active vehicle unloads is exactly equal to the load the passive vehicle receives; no load gets lost. The fundamental problem of load s. can thence be formulated as follows:

For each vertex with specified negative, zero, or positive demand, the difference between the total amount of load unloaded at the vertex by all active vehicles visiting it and the total amount of load received at the vertex by all passive vehicles visiting it must be equal to the specified demand.

There are three subtypes of load s.:

(a) Fixed load s.
This is the case when the direction and the amount of load transfer between two vehicles is fixed in advance, for example, when the application context requires that during a transshipment, the active vehicle always unloads completely.
(b) Discretized load s.
This is the case when there is a finite number of possible amounts of load that can be delivered, collected, or transferred.
(c) Continuous load s.
In this case, the amount of load that can be delivered, collected, or transferred may be any real number between zero and the respective obvious upper bound.

(v) *Resource synchronization*
This is necessary when different vehicles compete for common, scarce resources. In the VRPTT, the use of TLs is limited. Different vehicles compete for the possibility to perform a load transfer to a certain passive vehicle at a certain location at a certain point in time. The fundamental problem of resource s. can accordingly be formulated as follows:

The total consumption of a specified resource by all vehicles must be less than or equal to a specified limit.

Remarks on the above classification follow.
The above classification is neither a partition nor a covering of the space of all possible types of synchronization, and concrete examples of synchronization requirements may be subsumed under more than one of the above types. In particular, all types can be *modelled* as resource s., due to the immense generality and flexibility of the resource concept. However, the classification captures significant aspects of synchronization and allows a structured view of synchronization as well as a structured review of the pertinent literature.
Defining the tasks is a matter of perspective, a modelling decision. An underlying application may only suggest one definition or other. For example, Bredström/Rönnqvist [20] describe an application in homecare staff scheduling, where two nurses must visit a disabled person at the same time for lifting purposes or with a fixed offset to apply medicine after a meal. One option is to say that a task consists of the visits of two nurses. The other option is to say that each visit of a nurse is one task. In the following, it will become clear from the context what a task is in each application.

Operation and movement s. are also denoted *space-time s.* The distinction between operation and movement s. always depends on how the underlying graph or network is constructed. Nevertheless, this distinction is made here to reflect the real-world situation.
Space-time and load s. are interdependent if the time needed for a load transfer depends on the amount of load transferred (*load-dependent load transfer times*), as in the VRPTT. If splitting of loads is allowed, task s. requires load s.
Synchronization is not restricted to be performed between only two vehicles. The principle extends in a straightforward manner to three or more vehicles/objects and to transshipments in more than one direction at the same location at the same time. See, for example, Recker [107], Bürckert et al. [24], Rivers [108], Li et al. [86], Cheung et al. [30].

4 Modelling and algorithmic issues

VRPMSs are generalizations of VRPs. As such, the modelling and algorithmic tools developed for VRPs can basically be used for VRPMSs, too, but, as this survey will show, additional modelling and solution efforts are in most cases necessary to solve VRPMSs.
Seen from a *modelling perspective*, it is sometimes convenient to introduce complex networks for VRPMSs. For example, vertices do not need to directly correspond to a real-world location only, but may instead represent points in space and time, or even multidimensional objects such as space-time-operation-vehicle combinations and the like. Several solution approaches for VRPMSs discussed below are based on canny network representations.
When developing MIP models for VRPMSs, all types of s. can be represented as constraints. The unified model by Desaulniers et al. [47] provides a suitable framework for representing all VRPMSs discussed in this survey.
Seen from an *algorithmic perspective*, the standard exact approach for solving vehicle routing and scheduling problems formulated as MIPs is the branch-and-cut-and-price (BCP) principle, that is, the combination of cut and column generation embedded in branch-and-bound (Desaulniers et al. [48]). In column generation terminology (see, for example, Lübbecke/Desrosiers [91]), the synchronization constraints are basically *coupling* or *linking* or *joint* constraints, which go into the master problem, and which provide dual prices guiding the generation of new variables/columns by the solution of the subproblem. The solution of the master problem is mostly not too difficult for VRPs. The difficulty lies in solving the sub- or pricing problem, which has the structure of an elementary shortest path problem with resource constraints (ESPPRC, see Irnich/Desaulniers [79]). This problem is usually solved by a so-called labelling algorithm based on dynamic programming. Desaulniers et al. [47] give properties the subproblem has to fulfil so that such a labelling algorithm can be applied. As pointed out in [52], unfortunately, these properties are not fulfilled for the VRPTT subproblem (and also not for other VRPMSs). Other approaches for the exact solution of the ESPPRC subproblem have not been successful up to now (Jepsen et al. [81]). Therefore, the exact solution of many types of VRPMSs, most notably the VRPTT, is an open research topic.
The standard heuristic approach for solving large-scale real-world rich VRPs is based on local search (Funke et al. [58]) and/or large neighbourhood search (Ropke [109]) embedded in a metaheuristic (Glover/Kochenberger [63]). Basically, local search procedures for VRPs exploit the fact that the routes are independent of one another, so that changes to one route (or two routes in the case of a swap move etc.) do not affect other routes. The interdependence problem encountered in VRPMSs precisely means that routes in VRPMSs *are* affected by changes to other routes. This is relevant for the feasibility of other routes as well as the objective function value of a solution. A change in one route may make all other routes infeasible, so the evaluation of a move may require checking the feasibility of all other routes. If, as usual in VRPs, the objective is to minimize the overall distance travelled or the number of vehicles used, the evaluation of the overall objective function remains easy also for VRPMSs, because the contribution of each route to the objective function remains independent of other routes. If, however, the objective

is, for example, to minimize the maximal route duration or the duration of the execution of the complete route plan, then to recompute the objective function value after a move in a standard VRP still requires only the recomputation of the schedule of the modified route(s), whereas in VRPMSs, the interdependence problem may require the rescheduling of all routes.
A fundamental observation in standard VRPs is that a given solution in form of a set of vehicle routes, or, in other words, a vehicle flow, completely determines the path each request takes, be the latter a simple demand or supply request in a classical VRP or a pickup-and-delivery request. This is due to the fact that a request is transported by exactly one vehicle, and that only this vehicle visits the corresponding request location(s). When transshipments are possible, this is no longer the case, because the vehicle picking up a request need not necessarily transport it to the depot/delivery location. In single-commodity problems, that is, when a homogeneous, substitutive good is to be transported, as in the VRPTT, this is not an issue. In multi-commodity problems, that is, pickup-and-delivery problems where each request consists in the transport of a unique, non-substitutive commodity such as a parcel or a letter between a dedicated pickup and a dedicated delivery location, though, the problem of determining *request leg sequences* arises. For example, if request r is to transport some good from location r^+ to location r^-, and if it is possible to transship the request at a TL l, then it must be decided whether to transport the request over one leg $r^+ \to r^-$ with one vehicle, or over two legs $r^+ \to l$ and $l \to r^-$ with two vehicles. (Note that, if a leg for some request r is from location l_1 to location l_2, this does not mean that the vehicle transporting r from l_1 to l_2 drives directly from l_1 to l_2: After picking up r at l_1, the vehicle may visit an arbitrary number of locations before reaching l_2. However, r will stay on k from l_1 to l_2.) Using the second possibility obviously induces a dynamic time window for the second leg, since the second leg cannot be performed before the first one is finished. Consequently, if there is a change in the route performing the first leg before visiting l or in the route performing the second leg after visiting l, the respective other route is affected. This effect may further propagate and may, in the worst case, affect all routes. In particular, such a change may be the insertion or the removal of a leg sequence or a leg in insertion heuristics.
The determination of leg sequences need not be the first step in a heuristic, nor need they be determined explicitly at all. In MIP approaches, the decision variables must be such that request leg sequences are either explicitly modelled by decision variables or can be reconstructed from a solution.
Since the VRPTT contains all aspects of s. identified in Section 3, a successful exact or heuristic solution procedure for the VRPTT would provide a general procedure for VRPMSs. However, no powerful solution algorithm for the VRPTT yet exists. Moreover, it is probable that more specialized algorithms for VRPMSs with fewer synchronization requirements are easier to develop and lead to better solution quality and shorter computation times. Nevertheless, two research goals for the near future concerning VRPMSs are (i) the development of an exact branch-and-cut-and-price algorithm for the VRPTT capable of solving larger problems than the tiny instances described in [52] and (ii) the development of a metaheuristic combining large neighbourhood search and local search capable of solving real-world VRPTTs. Therefore, with respect to modelling and algorithmic approaches surveyed in this paper, the focus will be on such methods. The literature survey below will pay special attention to how the abovementioned two central issues, the solution of the pricing problem in BCP algorithms, and the solution of the interdependence problem in (local search) heuristics, are addressed. Other potential promising ways for solving VRPMSs and their potential advantages and drawbacks will nevertheless be outlined.

5 Literature survey

As its title implies, this paper is a survey on synchronization in *vehicle routing* problems. Therefore, problems which are not VRPs are not considered, and neither are VRPs without multiple synchronization constraints.

With respect to applications of VRPMSs, or, put differently, with regard to the *causes* for the existence of multiple synchronization constraints in a problem, four main types were identified:

(i) the possibility of splitting the pickup or the delivery of load at a customer between several visits by several vehicles

(ii) the possibility or requirement of transshipment of load or transfer of persons

(iii) the requirement of simultaneous presence of vehicles at a location to render a service

(iv) the existence of non-autonomous vehicles

With respect to the types of s. that appear in a problem in addition to task s., a considerable number of applications and publications was found for the following:

(i) Load s.

(ii) Resource s.

(iii) Operation s.

(iv) Movement s.

(v) Operation and load s.

(vi) Movement, operation, and load s.

The subsequent literature review is structured by subtype of s. Although the literature on VRPMSs is not yet as extensive as that on the VRP or the VRPTW, it is beyond the scope of this survey to give a detailed review of all relevant contributions. Therefore, in what follows, problems with load and resource s. are only briefly discussed, before describing in greater detail modelling and solution approaches for the different subtypes of operation and movement s. To avoid redundancies, problems of types (v) and (vi) are described in the section on movement s. To provide a unified treatment, the Appendix contains a glossary of terms, a summary of abbreviations and the notation used in the rest of the survey unless otherwise specified.

5.1 Load synchronization

VRPMSs with exclusively task and load s. are known in the literature as *split delivery VRPs*. In these problems, it is allowed that several vehicles visit a customer, each delivering (collecting) a part of the customer's demand (supply). Load s. is necessary at the customer vertices, even though no transshipments are allowed. Due to the limited size of this survey and the fact that the problem is well-studied (at least in comparison to other VRPMSs), for more information on the split delivery VRP or pickup-and-delivery problem (PDP), the reader is referred to Hooker/Natraj [74], Chen et al. [29], Archetti/Speranza [10], Nowak et al. [99], Schönberger et al. [114], Desaulniers [46], Derigs et al. [45], and Hennig [70].

5.2 Resource synchronization

Resource s. was introduced under the name *inter-tour resource constraints* in Hempsch/Irnich [69]. There, a generic model for representing rich VRPs with resource s. is developed. This model is based on the unified framework by Irnich [78] and uses the giant-route representation (Christofides/Eilon [33], Funke et al. [58]) and the concept of resource-constrained shortest paths (Irnich/Desaulniers [79]). The innovative idea is that the giant route is considered as one single, resource-constrained shortest path. By doing so, efficient solution procedures for local search developed for VRPs without resource s. can be used also for VRPs with resource s.
Examples of resource s. abound. Hempsch/Irnich [69] mention a limited number of docking stations at depots, a limited number of routes with certain properties such as distance or duration,

time-varying sorting capacities at mail-sorting centres, and the allocation of a limited fleet to several depots.
Two exemplary contributions are Ebben et al. [53] and El Hachemi et al. [54]. Ebben et al. [53] study a problem of dynamic scheduling of automated guided vehicles (AGVs) at an airport. Origin-destination transport requests have to be fulfilled by AGVs capable of performing one request at a time. There are several scarce resources to be considered, namely, the number of available AGVs, the number of docks for loading and unloading cargo, parking places for currently unused AGVs or AGVs waiting for free docks, and cargo storage space. The authors develop a heuristic serial scheduling procedure based on sequential capacity checking: A feasible partial schedule is extended by sequentially selecting an unscheduled activity according to a specified priority rule. However, despite the simplicity of its underlying basic idea, the procedure is quite sophisticated. El Hachemi et al. [54] study an application in the context of forest management. Vehicles have to transport wood from forest areas to mills. For loading the wood, there is one loading machine at each area, capable of loading one vehicle at a time. Hence, if more than one vehicle is present at an area at a time, at least one vehicle has to wait. The objective is to minimize the sum of waiting times for vehicles and loading machines. The authors use a combination of constraint and integer programming to solve their problem.
It must be noted that it is highly difficult to find publications considering VRPs with resource s., because it usually cannot be deduced from the title or the keywords whether or not resource s. is relevant in a paper. The cited references were found by chance. Resource s. is also not the focus of this survey. For further study of VRPMSs of this class, the reader is referred to Hempsch/ Irnich [69].

5.3 Pure spacial operation synchronization

The applications described in this section all consider transshipment possibilities. This obviously introduces interdependencies between routes and makes operation s. non-trivial even when the time aspect is neglected.
The following table gives an overview of the papers considered in this section.

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[111]	School bus routing	Buses	Pure spacial operation, fixed load	–	Heuristic hierarchical decomposition
[11]	Pickup-and-delivery of persons	Small buses	Pure spacial operation, fixed load	Arc	Standard MIP solver
[108]	Bitumen delivery, mid-air refuelling of aircraft, PDP with transshipments, school bus routing	Abstract, autonomous vehicles	Pure spacial operation, continuously split load	–	Cluster-first-route-second, local search
[3]	CARP	1 task & 1 support vehicle	Pure spacial operation, fixed load	Arc	Branch-and-cut
[4]	CARP	1 task & 1 support vehicle	Pure spacial operation, fixed load	Arc	Branch-and-cut, heuristic route-first-cluster-second
[66], [103]	N-echelon VRP	Task & support lorries	Pure spacial operation, fixed load	Arc	Branch-and-cut
[104], [105]	2-echelon VRP	Task & support lorries	Pure spacial operation, fixed load	Arc	Branch-and-cut
[65]	N-echelon LRP	Task & support lorries	Pure spacial operation, fixed load	Path	–
[5]	N-echelon LRP	Task & support lorries	Pure spacial operation, fixed load	Arc	Standard MIP solver
[40]	2-echelon VRP	Task & support lorries	Pure spacial operation, fixed load	–	Hierarchical decomposition, cluster-first-route-second, multi-start local search
[98]	2-echelon LRP	Task & support lorries	Pure spacial operation, fixed load	–	Hierarchical decomposition, hybrid GRASP and evolutionary/iterated local search

(continued on next page)

(continued from previous page)

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[15]	2-echelon LRP	Task & support lorries	Pure spacial operation, fixed load	–	Hierarchical decomposition, tabu search

Table 1: Papers on pure spacial operation synchronization

Russell/Morrel [111] present an early example for the introduction of transshipments in the context of school-bus routing. The problem is decomposed: First, routes from student residences to the TLs (two large schools) are computed by means of a modified savings heuristic and the M-tour procedure developed by the first author in [110]. Each student may be transported to any of the two TLs. Afterwards, the second stage, that is, the transport from the TLs to the final destinations, is 'relatively straightforward and can be done by hand' (p. 61).

Baker et al. [11], in an early contribution, describe an MIP for a passenger transport system with several depots and one vehicle each. The passengers are to be transported to a specified depot. The routes of the vehicles must start and end at their respective depot. Passengers are allowed to change vehicle at one TL, which must be (i) selected out of a set of several potential locations, and (ii) visited as the penultimate stop on each vehicle's route. The vehicle capacity is maintained by requiring that each vehicle not pick up more passengers than its capacity before visiting the TL, and by the fact that the overall number of passengers for each depot does not exceed the capacity of the respective vehicle.
The model uses two types of binary variable, three-index vehicle-arc flow variables and binary TL selection variables. The time aspect is considered implicitly: The vehicle which arrives first at the TL simply waits for the other vehicles. This, together with the fact that there is exactly one TL, ensures operation s.
The authors perform computational experiments with random test instances involving two depots. They compare the solution to the MIP with (i) the solution to the isolated situation, in which each vehicle is allowed to pick up only the passengers that must be transported to the vehicle's own depot (that is, two separate TSPs must be solved), and (ii) a model where it is allowed that the vehicles also transport passengers destined for the other depot, and where each vehicle therefore visits the other depot as the penultimate stop on its route, but where no passenger transfer is allowed. The results show a considerable potential for cost savings when the model with transshipment possibilities is used.

Rivers [108] develops heuristic solution procedures (construction and improvement, but no metaheuristics) for VRPMSs with pure spacial operation and continuously split load s. with a fleet of homogeneous, autonomous vehicles in Euclidean and rectilinear static and dynamic systems with coordinates for the vertices and without time windows. Each vertex represents a customer location with a certain demand. Load transfers between vehicles are possible at each customer location, but nowhere else.
The fundamental idea is to use s. as an *improvement* procedure which is based on an initial feasible solution obtained without s. by sweep-like cluster-first-route-second methods, followed by standard vertex and arc exchange local search improvement procedures. Afterwards, promising potential TLs are identified. This is done according to three basic approaches: (i) every vertex of every route is checked for whether any other route comes close to it; (ii) for every combination of routes, a vertex is sought which is close to at least two routes; (iii) vertices that are close to an intersection of two routes are sought. As in [11], the time aspect is implicitly considered by the assumption that the vehicle which arrives first at a TL waits.
The author performs computational experiments with sets of randomly generated test instances and shows the general usefulness of synchronization in such settings, although the potential for synchronization differs considerably between instances.

It is an open research question how the results obtained by Rivers can be transferred to discrete networks with time windows.

Amaya et al. [3], in the context of road marking, consider a capacitated arc routing problem (CARP) with one task and one support vehicle, both stationed at a central depot. The support vehicle can reload the task vehicle (with marking paint) at any road junction, and it must return to the depot after each reloading.
To model pure spacial operation s., the authors construct a network with arcs representing road segments between vertices representing road junctions and two anti-parallel artificial arcs between each vertex and the depot, representing a trip of the support vehicle. By this ingenious idea, operation s. is implicit in the network: Only one route, to be precise, directed walk, is computed. This walk consists of arcs for the road segments traversed by the task vehicle for either marking or deadheading, and, if a reloading is performed at a vertex i, of two antiparallel artificial arcs $(i, depot)$ and $(depot, i)$. From this walk, the routes of the two vehicles can easily be reconstructed: The task vehicle uses all non-artificial arcs in the sequence specified by the walk, the support vehicle visits the road junctions specified by the artificial arcs in the walk in the sequence the arcs appear in the walk.
The authors present an IP formulation based on the network using two types of variable: x_{na}, which indicates the number of times arc a is traversed after the nth and before the $(n+1)$st reloading, and y_{na}, which equals one iff a is serviced after the nth and before the $(n+1)$st reloading. The formulation contains 10 types of constraint, none of which needs to address operation s. The authors perform computational experiments with a basic cutting-plane algorithm, solving instances representing road networks with up to 70 vertices and 600 arcs.

Amaya et al. [4] consider an extension of the problem studied by Amaya et al. [3]. The difference is that the support vehicle does not need to return to the depot after each reloading. Therefore, two problems have to be solved simultaneously: A VRP for the support vehicle and a CARP for the task vehicle. This makes the new problem more difficult to solve than the previous one. Nevertheless, the authors succeed in constructing a network which again models pure spacial operation s., and an IP model based on this network. Both the network and the IP are extensions of their counterparts from the earlier paper. The network contains four types of arc: (i) arcs representing road segments, and three sets of artificial arcs connecting (ii) the depot to each vertex at zero cost, (iii) each vertex to the depot at zero cost, and (iv) all pairs of vertices with one another with a cost equal to the length of a shortest path between the respective road junctions. The first three sets are to be used by the task vehicle, the fourth one by the support vehicle. The formulation uses the same variable types as in the previous work, contains 12 types of constraint, and again, none of these needs to address operation s.
Additionally, Amaya et al. develop a heuristic for the problem based on the route-first-cluster-second principle. Due to the construction of the network, the heuristic does not need to address operation s. and is therefore not described here.
The authors perform computational experiments with the IP model and the heuristic. The results show that large instances cannot be solved with a direct implementation of the IP model in a standard MIP solver. The heuristic obtains, in short computation time, solutions with a small gap to the lower bound provided by the IP model.

Gonzalez Feliu et al. [66] and **Perboli et al. [103]** formally introduce the class of *multi-echelon* (or *N-echelon*) *vehicle routing problems* and are the first to use these terms. The basic idea behind this problem class is that customers are not delivered directly from a central depot, but via N legs in an N-stage distribution network. An N-stage distribution network contains $N+1$ levels of location. Echelon or stage $n \in \{1, \ldots, N\}$ considers transports from location level $n-1$ to n. For each stage n, there are dedicated vehicles which can only visit the locations defining stage n. This means that only the vehicles of stage N are task vehicles, that is, are

allowed to visit customers; all other vehicles are support vehicles. All vehicles are autonomous. Load transfers are only possible between vehicles of different stages. The difference to distribution network design problems lies in the fact that for each vehicle in the problem, a complete route is computed.
The authors discuss several variants of multi-echelon VRPs. They mention the existence of variants with pure spacial as well as exact simultaneous operation s. and operation s. with precedences. For load s., also different options exist, depending on whether or not load splitting on which echelon(s) is allowed and whether or not TLs may be visited by more than one vehicle for unloading.
The authors present an MIP model for the two-echelon case with one level-0 location v_0, capacitated level-1 locations, denoted by V_1, pure spacial operation s. and fixed load s. (the demand of each customer cannot be split, neither at the first nor at the second stage). The model uses the following types of variable: (i) x_{ij} indicating whether a first-level vehicle visits TL i directly before j, (ii) x_{lij} indicating whether a second level vehicle starting at TL l visits customer i directly before j, (iii) assignment variables z_{lj} indicating whether customer i is serviced from TL l, (iv) continuous q_{ij} load flow variables indicating the amount of load flowing from i to j on the first level, and (v) continuous q_{lij} load flow variables indicating the amount of load originating at TL l and flowing from customer i to j on the second level. Overall, there are 16 types of constraint. Operation and load s. at all vertices are achieved by flow conservation constraints for vehicles *and for load* and by constraints of the form

$$q_{ij} \leqq Q^1 x_{ij} \quad \forall\, i,j \in \{v_0\} \cup V_1, \tag{1}$$

where Q^1 is the capacity of each first-stage vehicle, to link the flow of vehicles and load on the first stage, and analogous constraints for the second stage.
The authors derive valid inequalities for their model and solve test instances with up to 5 TLs and 50 customers by means of a branch-and-cut algorithm using the derived inequalities.

Perboli et al. [104], [105] develop valid inequalities for the formulation presented in [66] and [103] and show the effectiveness of these inequalities by performing computational experiments with the test instances described in the latter paper.

Gonzalez-Feliu [65] studies the general N-echelon *location*-routing problem. The difference between the N-echelon VRP and the N-echelon LRP is that the latter, contrary to the former, considers fixed costs for opening a TL. The author surveys relevant literature also discussed in the present paper and presents a path-based MIP model. A path variable λ_p^k represents a feasible route within the stage that k belongs to. Additionally, binary z_l variables indicate whether location l is open, and q_l indicates the amount of load transported to location l. The model contains six types of constraint. Pure spacial operation s. is achieved by two sets of constraints. One set requires that the load transported to the locations of level n in stage n equal the load transported to the locations of level $n-1$ in stage $n-1$. The other set states that for each location l at level n, the load delivered equals the load picked up by all routes of stage $n+1$ that start at l.

Ambrosino/Scutellà [5] study multi-echelon location-routing problems. They develop three MIP formulations for the 3-echelon case, where for the first stage, only direct transports are possible. Routes are computed for the second and the third stage.
The first formulation essentially uses binary x_{ij}^k routing variables, y_{lj} assignment variables as in [66], z_l variables equalling one iff TL l is opened, and continuous load variables q_l indicating the amount of load flowing directly from the central depot to first-stage TL l and q_{ij}^k indicating the amount of load transported from level-1 location i to level-2 location j by vehicle k (this does not necessarily mean that i is visited directly before j on k's route). The first formulation

contains 21 types of constraint. Operation s. is ensured by the highly complex interplay of 6 different types of constraint and cannot be described in detail here.
The second formulation is based on the first one, where the q_{ij}^k variables are redefined to model the flow of load along each arc (i, j). This allows to replace 11 types of constraint by 8 new ones and to remove 3 types of variable.
The third formulation extends the first one by considering a multi-period planning horizon by adding a period index to each variable type and introducing an additional variable type for the stock levels at each location at the end of each period. (This is, in fact, already a situation with operation s. with precedences, where the precedences are ensured by the discrete time periods. As will be shown in Section 5.4, the idea of discretizing time to ensure operation s. with precedences is used by several authors.)
Computational experiments using a standard MIP solver are performed to validate the models.

Crainic et al. [40] present multi-start heuristics to solve the 2-echelon VRP with capacitated TLs and one level-0 location. Their basic idea is to separate the problem into two routing subproblems, one for each stage, and to further decompose the second stage into independent VRPs, one for each TL. An initial solution is computed by first solving the second-stage problem by assigning each customer to its closest TL, taking into account TL capacities. The result is then used as input for the first-stage problem. In this way, pure spacial operation s. and load s. are ensured. To improve given solutions, local search is performed within one neighbourhood (one customer from the given solution is assigned to a different TL). New solutions for the multi-start component are constructed by two stochastic rules specifying probabilities for the assignment of a customer to a TL and performing roulette-wheel selection. Computational experiments with the instances used in [66] yield promising results.

Nguyen et al. [98] also study the 2-echelon location-routing problem with capacitated TLs and one level-0 location. The authors develop a sophisticated hybrid metaheuristic that alternates between a greedy randomized adaptive search procedure (GRASP) and an evolutionary/iterated local search (ELS/ILS) algorithm, using tabu lists. To compute initial solutions, the GRASP uses three heuristics, one based on the savings algorithm, a nearest neighbour heuristic, and a procedure that first builds subtours and inserts the best TL afterwards. All three procedures also decompose the problem into two stages, solve the second-stage problem first and use its solution as input to the first-stage problem.

Boccia et al. [15] present an ingenious tabu search heuristic for the 2-echelon LRP with capacitated TLs and several level-0 locations. The heuristic integrates two well-known heuristics for LRPs, namely, the nested approach of Nagy/Salhi [96] and the two-phase iterative approach of Tuzun/Burke [122]. The problem is decomposed into two components corresponding to location-routing problems for each stage. Each component is further decomposed into a capacitated facility location problem and a multi-depot vehicle routing problem. Also these authors use the solution of the second-stage problem as input to solve the first-stage problem.

Concluding remarks on pure spacial operation synchronization

The literature review shows that heuristics for two-stage problems (2-echelon VRP/LRP as well as Russell/Morrel [111]) use decomposition by stage as their central idea. When no time aspect is present, sequential consideration of stages is apparently the adequate strategy to obtain high-quality solutions.
Essentially, the 2-echelon VRP is a VRPTT as described in Section 2, but without trailers and with a fixed assignment of tractors and semi-trailers. However, the VRPTT comprises also the N-echelon VRP and LRP for arbitrary N. This is simply a matter of modelling the fleet and the structure of the network defining a VRPTT instance.

5.4 Exact operation synchronization

As the following table shows, the references discussed in this section consider a wide variety of applications.

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[76], [12]	Aircraft fleet routing and scheduling	Aircraft	Simultaneous operation	Arc, path	Branch-and-price
[20]	Homecare staff scheduling, planning of routes for security guards, forest management	Persons with different qualifications, cranes and forwarding vehicles	Simultaneous, deferred operation; operation with precedences	Arc	MIP-based heuristic
[86], [87]	Staff scheduling	Workers with different qualifications	Simultaneous operation	Arc	Heuristic: Parallel insertion & simulated annealing with indirect search
[52]	Raw milk collection	Lorries/tractors and trailers/semi-trailers	Movement en route, exact operation, continuously split load	Arc, turn, path	Branch-and-cut, Branch-and-price
[51]	Staff scheduling	Workers with different qualifications	Simultaneous operation	Arc, path	Branch-and-price
[55]	Pickup-and-delivery of swap-body platforms in long-haul road transport	Lorry-trailer combinations with capacity 2	Operation with precedences, fixed load, resource	–	Iterative, nested solution of capacitated non-bipartite matching, large neighbourhood search
[41]	City logistics modelled as 2-echelon VRP	Task & support lorries	Simultaneous operation, fixed load	Path	–
[42]	CARP	Large and small garbage collection vehicles	Simultaneous operation, fixed load	–	Route-first-cluster-second, local search
[43]	CARP	Large and small garbage collection vehicles	Simultaneous operation, fixed load	–	Construction by VND-CARP ([72]), improvement by VNS
[113]	Concrete delivery to construction sites	Heterogeneous task & support vehicles	Soft simultaneous operation, discretely split load	Arc	VNS, VLNS using MIP-based heuristic

Table 2: Papers on exact operation synchronization

Ioachim et al. [76] propose an exact branch-and-price approach for an aircraft fleet routing and scheduling problem containing so-called same-departure-time requirements: Subsets of flights to be flown on several days during a week have to depart at the same time every day.
The authors present a compact formulation from which the extended formulation used for branch-and-price is derived. The compact formulation is based on a network with one vertex for each task, that is, flight. Two vertices are linked by an arc iff an aircraft can perform the corresponding flights consecutively.
Let M be the set of groups of flights requiring operation s., let $m_i \in M$ indicate the group to which flight $i \in R$ belongs, and let R' be the set of all flights with same departure time constraints. The compact formulation contains binary x_{ij}^k variables and two types of time variable, t_i^k, indicating the departure of flight i, and t_{m_i}, indicating the time that elapses between a_i, the earliest departure time of any flight i in group m_i, and the actual departure of any flight in m_i. Exact simultaneous operation s. at different vertices, that is, guaranteeing that flights belonging to the same group depart at the same time, is then ensured by the two constraint types

$$a_i \sum_{(i,j)\in A} x_{ij}^k \leqq t_i^k \leqq b_i \sum_{(i,j)\in A} x_{ij}^k \quad \forall\, k \in F, i \in R, \tag{2}$$

which ensure that the t_i^k variables are zero if flight i is not performed by aircraft k, and

$$\sum_{k \in F} t_i^k - t_{m_i} = a_i \quad \forall\, i \in R' \subseteqq R. \tag{3}$$

Constraints (3) and the t_{m_i} variables remain in the master problem of the column generation formulation. If the time variables are continuous, this is sufficient to ensure operation s. However, this leads to a subproblem which is a shortest path problem with time windows and linear costs on the time variables at the vertices. If all these time costs are the same at every vertex, the problem can be solved by a standard labelling algorithm as demonstrated by Desaulniers/ Villeneuve [50]. In the application studied by Ioachim et al. [76], however, these linear costs are different at different vertices and may be positive as well as negative. This prohibits the solution of the subproblem with a standard labelling algorithm. For details on this issue, see Ioachim et al. [77]. In the latter paper, a special labelling algorithm is developed that is able to handle linear costs at vertices. Ioachim et al. [76] use this labelling algorithm for the solution of their pricing problems.
To ensure operation s. in the case where the time variables are discrete, the authors develop a dedicated branching scheme that makes use of branching on the time variables, an approach that was first used in branch-and-price algorithms by Gélinas et al. [59].
Finally, computational experiments with real-world data are performed. The problems are solved to optimality.

Bélanger et al. [12] study a problem similar to that of [76] in a similar application context where the same issues with respect to the solution of the pricing problem arise. Bélanger et al. also consider constraints for the s. of departure times in the master problem and use the algorithm by [77] to solve the pricing problems. Moreover, they develop specialized, rather involved branching strategies which are in part based on branching on the time variables.

Bredström/Rönnqvist [20] describe applications in homecare staff scheduling, where two nurses must visit elderly or disabled people at the same time for lifting purposes or with a fixed offset to apply medicine after a meal. According to the authors, similar s. requirements arise in the planning of security guards, where two guards have to inspect buildings at night, and in forest management, where mobile cranes must assist lorries in the loading of felled trees. The problems constitute generalizations of the VRPTW with vehicle-task compatibility requirements and both simultaneous and deferred exact operation s.
The authors develop an MIP model based on a network which, for each task requiring n vehicles, contains n vertices, each of which corresponds to the same physical location. They introduce binary x_{ij}^k variables and continuous t_i^k time variables indicating the point in time when vehicle k starts executing the task associated with i. Then, the following type of constraint, together with a constraint type equivalent to (2), ensures synchronization of vehicles:

$$\sum_{k \in F} t_i^k = \sum_{k \in F} t_j^k + \Delta_{ij} \quad \forall\, (i,j) \in R^{sim} \cup R^{def}, \tag{4}$$

where R^{sim} (R^{def}) is the set of vertex pairs that require simultaneous (deferred) visits, $\Delta_{ij} = 0$ for all pairs in R^{sim}, and $\Delta_{ij} > 0$ the time period that must elapse between the visits to i and j for all $(i,j) \in R^{def}$. Besides, replacing the '=' relation in (4) with '$\leqq$' guarantees that i is visited at least Δ_{ij} time units before j, in other words, this guarantees operation s. with precedences.
The authors perform computational elements with test instances of up to 80 tasks, 16 vehicles, and $|R^{sim}| = 8$. They compare the direct solution of their formulation by a standard MIP solver with the solution by a heuristic solution approach that iteratively solves restricted MIP problems to improve the best known feasible solution. The problems are restricted with respect to the vehicles that are allowed to perform a task.

The results show that (i) the developed model is not much harder to solve than a VRPTW without multiple synchronization constraints, and that (ii) an increase of the proportion of visits to be synchronized does not make the model harder to solve, which is surprising.

Li et al. [86], based on earlier work ([87]), introduce the manpower allocation problem with time windows and job-teaming constraints (MAPTWTC). In this problem, tasks have to be performed by workers. Each worker belongs to one of several qualification classes. Each task requires one or more workers of one or more qualification classes, takes a specified time to execute, and execution must begin during a given time window at a given location. All workers required for carrying out a task have to be present at the task location before execution can begin and may leave the location only after execution is finished. Hence, the problem requires exact simultaneous operation s. The objective is to minimize the weighted sum of the total number of workers needed to fulfil all tasks and the total travel time of all workers.
The authors present an MIP formulation based on a network with one vertex per task, using binary x_{ij}^k variables and continuous t_i variables indicating the when execution of task i starts. The use of one time variable per vertex, together with the following constraint type, ensures exact simultaneous operation s. (t_{ij} is the travel time from i to j):

$$x_{ij}^k = 1 \Rightarrow t_i + t_{ij} \leqq t_j \quad \forall\, k \in F, (i,j) \in A \tag{5}$$

The authors describe two construction procedures and a simulated annealing heuristic to solve the problem. Both construction heuristics receive a permutation of the set of tasks and an initially empty set of workers' schedules as input. The permutation is either a random one, or one sorted by increasing task start times or increasing or decreasing task execution times. Then, a yet unfulfilled task j is selected from the permutation. In the first heuristic, it is checked which active workers, that is, workers having already a task assigned, can work on j as the last task on their schedule, respecting task time windows and worker qualifications. If not enough active workers are available, new ones are activated. This is repeated until all tasks are assigned.
In the second heuristic, when j is to be scheduled, an optimal subset of the set of active workers is computed by an enumerative algorithm such that the workers in the subset have a common time interval that allows execution of j. The subset is optimal in the sense that it requires a minimal number of new workers to be activated. Additional workers are activated as necessary. Hence, in both heuristics, operation s. is achieved by (i) determining a set of workers and a position for j in the workers' task sequences, and (ii) setting the start time of execution of j to the earliest time when all workers assigned to j can start working on j.
As local search operators in the simulated annealing metaheuristic, two neighbourhoods operating *on a permutation*, and not on a given feasible solution, are used. This is called *indirect search* (see [44]) and constitutes an elegant way of addressing the interdependence problem. The first neighbourhood reverses the order of a subset of the tasks within a permutation, the second one exchanges the position of two subsets in a permutation, leaving the order within the exchanged subsets unchanged.
The authors perform computational experiments with a set of test instances they developed. The results obtained with the heuristic are close to the lower bounds computed with a standard MIP solver.

Dohn et al. [51] study also the MAPTWTC introduced by [86], but they specify a maximal number of teams *allowed* to perform a task and pursue the objective of maximizing the number of assigned tasks over all teams. This means that under-fulfilling of tasks is allowed; if some tasks are performed by fewer teams than possible, this does not make a solution infeasible.
Dohn et al. describe a branch-and-price approach based on the compact formulation introduced by [86]. They discretize the time by making the t_i variable indicating the beginning of execution of task i a general integer variable which can take values from a finite interval $[a_i, b_i]$. By means of an ingenious and quite involved reformulation of the problem, the authors obtain a restricted

master problem that does not contain the t_i time variables. Operation s. is then ensured by branching on the time variables, similar to [76].
The advantages of this procedure over the approach taken in [76], where the time variables remain in the master problem, are that (i) no non-binary coefficients exist in the master problem (thus making the solutions of the LP relaxation less fractional on average), and that (ii) the subproblem becomes a standard elementary shortest path problem with resource constraints as in a VRPTW and can be solved by a standard labelling algorithm (a much more involved algorithm such as the one used in [76], described in detail in [77], is not necessary).
The authors perform computational experiments with real-world instances of up to 20 teams and 300 tasks and are able to solve most instances to optimality.
In the case of deferred exact operation s., it is possible to introduce one vertex for each sub-task, that is, for each visit, and to have one time variable t_i for each vertex i. If two visits i and j are required to fulfil a task and j must be Δ time units after i, t_j can be replaced by $t_i + \Delta$. This means that, in effect, still only one time variable per task is necessary also in this case, so that the approach of Dohn et al. should be applicable. It is an open question, though, whether a reformulation similar to the one developed by Dohn et al. exists for the case of operation s. with precedences.

Feige [55] and **Werr (2007, personal communication)** describe a pickup-and-delivery problem with time windows (PDPTW) in the context of cooperative operational planning of long-distance linehaul road transports in Germany. Requests of different forwarders have to be transported in swap-body platforms between local depots. Each request has a capacity requirement of 0.5, 1 or 2 swap-body platforms, and each vehicle can transport 2 swap-body platforms at a time. The planning task comprises options for meet-and-turn transports as well as for transshipments and a large number of side constraints, among them 'visual attractiveness' of routes, precedence constraints for requests due to fixed agreements between the forwarders, limited capacities at TLs, and location-dependent load transfer times. The problem requires exact simultaneous operation s. as well as operation s. with precedences and fixed load s.
The problem is solved by a constructive heuristic followed by large neighbourhood search. The planning horizon is 24 hours, and first, time is discretized into 15- or 30-minute intervals. The constructive heuristic then consists of three steps: (i) pairwise grouping of requests to fill one swap-body platform, (ii) pairwise grouping of swap-body platforms to yield one full vehicle load, (iii) pairwise grouping of full loads to short routes and of short routes to longer ones. All three steps are modelled and solved as a capacitated non-bipartite matching problem. Each step may be performed several times before the next step or the large neighbourhood search starts. For example, if in step (ii), two requests $r_1^+ \to r_1^-$ and $r_2^+ \to r_2^-$ consisting of one swap-body platform each and with $r_1^- = r_2^-$ are paired at TL l, this means that the first request is transported from r_1^+ to l and from there to r_1^-, and the second request is transported from r_2^+ to l and from there to $r_1^- = r_2^-$. This leads to two new requests $r_1^+ \to l$ and $r_2^+ \to l$ for step (ii) and one complete load $l \to r_1^-$ for step (iii).
The matching problem is modelled and solved as an MIP using binary $x_{r_1 r_2 l t}$ variables which equal one iff requests r_1 and r_2 are paired using TL l in period t, and y_r, which equal one iff r is not paired with any other request. Moreover, a heuristic is developed which, in each step, enumerates all potential pairwise groupings, taking into account relevant options for the TLs. Due to the large number of real-world constraints, the technical details of the heuristic are extremely involved and cannot be described here. Basically, all side constraints, including the temporal constraints for operation s., have to be checked for each potential pairwise grouping in each step to determine its feasibility, and the synchronization complexity lies in determining this feasibility.
In the large neighbourhood search, existing routes are destroyed according to the following two criteria: (i) overall route cost and (ii) distance over which at most one loaded swap-body platform is transported. The selection of the routes to ruin contains a random component; routes with

high overall cost and a high distance driven partially empty are selected with higher probability. Routes are recreated by the constructive heuristic just described.
Computational experiments show that a solution of the MIP with a standard solver is too time-consuming for instances of realistic size (1,200 requests), whereas the computation time of the heuristic is short enough for operational planning and the solution quality is excellent.
The algorithm has been included into the commercial decision support system CARGO$_{plan}$ and was successfully applied to real-world instances at customers.

Crainic et al. [41] describe an extended version of the 2-echelon VRP in the context of city logistics. The models they study consider exact simultaneous operation s. of the dedicated support vehicles and the task vehicles at the TLs (a support vehicle may only arrive at a TL when there are enough task vehicles to receive the complete load of the support vehicle; vehicles must not wait; load cannot be stored at TLs, the latter really deserve their name in this case). The authors develop a model using path variables, based on a time-discrete network where there is one vertex for each pair (TL, time period). The time periods have a length of between 15 and 30 minutes. There are three types of path variable: One each for the paths/routes of the support and task vehicles, and one for the path each customer request takes. This is necessary, because the goods to be transported are not substitutive. Operation s. at the TLs is achieved by two types of constraint. The first one specifies that the respective numbers of support and task vehicles using a TL in any given time period t are equal to those that arrive at t plus those that have arrived before but have not yet left the location. The second one states that the respective numbers of support and task vehicles present simultaneously at a TL are provided by the flow of freight imposed by the demand itineraries. Load synchronization is trivial: The vehicle that visits a certain customer receives its complete demand from the support vehicle that transports this demand to the chosen TL.
The authors state that theirs is 'fundamentally a *modelling* paper ... detailed algorithmic developments are beyond the scope of the present work' (p. 433). Nevertheless, they propose a heuristic hierarchical decomposition consisting of a model for the routing of the support vehicles and a model for the routing of the task vehicles. The former model is supposed to receive as inputs the possible allocations of customers to TLs and an estimate of the costs of servicing each customer from its associated TL. The authors claim that this information is 'relatively easy to obtain' (p. 444). For the latter model, a further heuristic decomposition into the sequential solution of a vehicle routing and a network flow component is proposed. The vehicle routing component solves, for each (TL, time period) pair, a VRPTW for the customers associated with the respective pair. The flow component solves a network flow problem to provide the (TL, time period) pairs with a sufficient number of vehicles. Overall, the paper is very long, the presented models are extremely involved, and the necessary notation is highly burdensome.

De Rosa et al. [42] describe a CARP with transshipments in the context of waste collection. In their problem, a set of small vehicles must collect the supply at the required edges and transport it to a TL, where the supply is transferred to a set of large vehicles and transported directly to a final location. The objective is to design routes for the small and sequences of direct trips for the large vehicles such as to minimize the overall travelled distance, while fulfilling the following requirements: (i) In each transshipment process, a small vehicle transfers its complete load into one large vehicle in constant time and (ii) the routes of all vehicles must begin at time zero and end at time T. The first requirement makes load s. trivial, the second one mitigates the timing aspect of operation s., as the optimization potential given by the possibility to defer route start times of small vehicles to reduce waiting at the TL is ignored.
The authors develop a lower bound as the sum of a lower bound on the traversal cost of all required edges and a lower bound on the total cost of a schedule for the large vehicles. Moreover, they develop a tabu search heuristic for the problem. This heuristic first computes an initial solution by (i) solving a rural postman problem and cutting the solution into routes for the small

vehicles according to a procedure developed by Hertz et al. [71], and (ii) computing schedules for all vehicles by a greedy heuristic. The greedy heuristic works as follows: Large vehicles are loaded one at a time. If no large vehicle is available, arriving small vehicles must wait. Upon arrival of a large vehicle, the small vehicles are unloaded according to a FIFO policy. A large vehicle leaves the TL when it cannot accommodate the load of the next small vehicle to be unloaded. The neighbourhoods used in the tabu search are moving a required edge (i) to a different segment of the current solution, (ii) to a newly created route segment, or (iii) to a dummy route containing no required edge.
Due to the interdependence problem, it is computationally costly to maintain feasible solutions during the complete tabu search. Moreover, the evaluation of the influence of a move on the overall objective function value cannot be computed efficiently. To overcome this, the authors proceed similar to the unified tabu search heuristic developed in Cordeau et al. [37] and allow feasible as well as infeasible solutions. Infeasible solutions are penalized in the objective function. The authors use three penalty terms for violation of (i) vehicle capacity and duration of routes of (ii) small and (iii) large vehicles. The penalty values are weighted by dynamic self-adjusting parameters. Additionally, the authors derive sophisticated but easy-to-compute approximations for the objective function.
The authors perform computational experiments with 102 test instances containing up to 50 vertices, 97 required edges, 3 small, and 5 large vehicles. The results show an average difference between lower and upper bound of only 6.7 per cent for the largest instances.

Del Pia/Filippi [43] also study a CARP with transshipments in the context of waste collection. In the problem, there are two types of autonomous vehicle, small and large, and load transfers are allowed from the small vehicles to the large ones.
The authors develop a variable neighbourhood descent (VND) heuristic based on a local search procedure for the CARP proposed by Hertz/Mittaz [72]. The neighbourhoods used in the VND approach are inter-route edge exchanges. No MIP model is developed. Computational experiments with real-world instances are performed.
Operation s. is ensured as follows. First, the total collection amount of the small vehicles and the part of the capacity of the large vehicles reserved for transshipments from the small vehicles are fixed to sensible values proposed by experienced practitioners. These values may also be seen as parameters for what-if and sensitivity analyses. Then, a route plan with a concrete exact schedule for each vehicle is computed that serves all edges of the network with the available vehicles without considering any transshipments. Only after that, transshipments are considered to get feasible routes. This step is performed sequentially: First, a small vehicle is selected for a transshipment. The small vehicle which needs to perform a load transfer at the earliest point in time is considered first. In this way, only the schedules of passive, that is, large, vehicles are affected, but not the schedules of the other active, that is, small, ones, and interdependencies are reduced. For the small vehicle selected for load transfer, a part of its route is determined during which a transshipment is reasonable (because the vehicle has already collected some load) and necessary (because at the end of this partial route the vehicle is full). Then, a large vehicle and a vertex on its route are selected such that the total increase in the duration of both routes, when performing a load transfer at this vertex, is minimal. This means that only the routes of small vehicles are changed, which is suboptimal but facilitates the necessary computations considerably. The route plan is updated accordingly and the procedure repeats until all routes are feasible.

Schmid et al. [113] consider the problem of concrete delivery from several plants to construction sites by heterogeneous vehicles. The demand at each site exceeds the capacity of the largest vehicle, so several deliveries are necessary to fulfil a task. Only one vehicle at a time can deliver concrete at a site. Moreover, it is necessary that the flow of concrete at a site, once started, be almost uninterrupted. This means that when a vehicle delivering to a site is finished, the next

one should already be present. In addition, a vehicle can only load concrete for one site at a time, so it always delivers its complete load on each visit to a site and then returns to a plant to reload. The loading and unloading times are independent of the amount loaded or unloaded. Finally, there are support vehicles needed at some sites to assist in unloading. These vehicles must be present at the site during the complete delivery process and hence must be the first ones to arrive and the last ones to leave.
The authors develop an MIP model using five types of binary and three types of continuous time variable. Lower and upper bounds for the number of deliveries to a site are computed. These determine the number of variables. The most important binary variables are (i) $x^k_{i_1 n_1 i_2 n_2}$ equalling one iff vehicle k performs delivery n_2 at construction site i_2 directly after performing delivery n_1 at site i_1, (ii) y^k_{in} equalling one iff vehicle k performs delivery n at site i, and (iii) z_{in} equalling one iff the nth delivery at site i is performed at all. The two most important time variables are t^{start}_{in} and t^{end}_{in}, indicating the start and end of the nth delivery to site i respectively. Operation s. is ensured by constraint types stating that (i) the time between two consecutive unloading operations assigned to a vehicle is big enough to allow the vehicle to drive to the closest plant and reload there (this ensures that no two vehicles can unload at a site at the same time and that $t^{start}_{in} - t^{end}_{in-1} \geqq 0$ for all i and n) and (ii) the time between the beginning and the end of an unloading operation, that is, $t^{end}_{in} - t^{start}_{in}$, equals the unloading time of the vehicle performing the nth delivery at i. The requirement that the flow of concrete at a site, once started, be uninterrupted, is modelled as a soft constraint: The objective function contains a penalty term $c^{penalty}(t^{start}_{in} - t^{end}_{in-1})$ for each site i and each delivery $n > 1$. This can be called *soft exact deferred operation s.*
Load s. is ensured by a constraint type requiring that the sum of the capacities of the vehicles that have delivered to a site be greater than or equal to the site's demand.
The authors solve the problem by variable neighbourhood search (VNS) and very large neighbourhood search (VLNS) (Ahuja et al. [1]). An initial solution is created by generating a delivery sequence for each task, ensuring feasibility regarding task fulfilment, considering the tasks one by one. VNS is then performed on the resulting set of sequences. There is a shaking and an iterative improvement phase. The neighbourhoods used in the shaking phase basically replace a vehicle in a sequence by another vehicle. To evaluate the solution, a schedule for all vehicles is computed by 'forward and backward termination' inspired by the critical path method used in project management. Forward termination schedules every delivery as early as possible, taking into account vehicle availability. The result serves as input for backward termination, where every delivery is scheduled as late as possible. Unfortunately, the authors do not give full details on this ingenious idea. In the improvement phase, three local search operators are used. The first one tries to remove unnecessary vehicles from a sequence, the second one moves vehicles within one sequence, and the third one swaps vehicles between different sequences.
The basic idea of the VLNS approach is to use the MIP model where all variables are fixed according to a feasible solution obtained by VNS. In each VLNS step, some of the variables are unfixed, and the resulting MIP is solved by a commercial solver, using several additional cuts. The decision which variables to unfix is based on a hierarchical ranking of the variable types. The decision whether or not to execute a certain delivery, represented by the z_{in} variables, is at the top level of this hierarchy. The variables determining which vehicle to use for a certain delivery, y^k_{in}, constitute the second level. Most variables of these two types remain fixed in the VLNS step. All other variables are unfixed.
The authors solve a set of real-world test instances and obtain very good solutions.

Concluding remarks on exact operation synchronization

As a further approach for exact operation s., several authors propose to set a fixed visiting time (Grünert (2006, personal communication); Scheuerer, Sigl (2007, personal communication);

Bredström/Rönnqvist [20]). This, however, leads to the problem that the visiting times may be chosen badly and may hence lead to bad solutions.
An extension of this idea is to introduce an enlarged network with several duplicates for each original task vertex. In this enlarged network, each vertex has a single-point time window that corresponds to the point in time when execution of the task associated with the vertex begins (Desrosiers (2005, personal communication)). On the one hand, depending on the number of duplicates of each task vertex, this procedure creates large networks or coarse discretizations. On the other hand, the subproblems in a branch-and-price approach based on such a network are solvable by standard labelling algorithms.
Overall, it depends on the application context whether time can and should be modelled as continuous or discrete.

5.5 Operation synchronization with precedences

As can be seen from the following table, most papers discussed in this section consider dial-a-ride or pickup-and-delivery problems with transshipments. Persons or goods can be left behind at TLs by an unloading vehicle and be picked up some time later by a reloading vehicle. In other words, operation s. with precedences and fixed load s. are considered.

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[68]	Pickup-and-delivery of letter mail	Lorry-trailer combinations, aircraft	Operation with precedences, continuously split load	Arc	–
[95]	Pickup-and-delivery in long-haul road transport	Lorry-trailer combinations	Operation with precedences, fixed load	Path	Column generation
[38]	Pickup-and-delivery of persons	Heterogeneous vehicles	Operation with precedences, fixed load	Arc	Benders decomposition
[123]	Pickup-and-delivery at a cross-docking centre	Lorries	Operation with precedences, fixed load	Arc	Sweep-like construction, unified tabu search ([37]) embedded in adaptive memory procedure
[100]	General pickup-and-delivery	Abstract, autonomous vehicles	Operation with precedences, fixed load	–	Two-stage constructive, tabu search
[18]	Dynamic pickup-and-delivery in intermodal long-haul transport	Lorries, trains	Operation with precedences, fixed load	Arc	Sequential insertion
[80]	Newspaper distribution modelled as 2-echelon LRP	Task & support lorries	Operation with precedences, fixed load	–	Heuristics: Greedy spanning tree, hierarchical decomposition
[88]	Pickup-and-delivery of documents	Uncapacitated vans	Operation with precedences, fixed load	Arc	Tree search with explicit enumeration
[2]	Dial-a-ride	Small taxis and fixed-schedule buses	Operation with precedences, fixed load	–	Three-stage heuristic construction, local search
[116], [120]	Pickup-and-delivery of parcels	Uncapacitated vans	Operation with precedences, fixed load	–	Parallel best insertion, local search
[93]	Pickup-and-delivery of parcels	Uncapacitated vans	Operation with precedences, fixed load	–	Multi-start cheapest insertion, descent improvement
[67]	Dial-a-ride	Abstract, autonomous vehicles	Operation with precedences, fixed load	–	Route-first-cluster second approximation algorithm
[56], [57]	School bus routing	Buses	Operation with precedences	Arc	Greedy heuristic & local search, branch-and-cut

Table 3: Papers on operation synchronization with precedences

Grünert/Sebastian [68] study a pickup-and-delivery problem with transshipments. To consider the possibility of transshipments, the authors model the problem as a type of multi-commodity network flow problem, where flows for requests as well as for vehicles are computed. They abstract from a direct assignment of each request to a concrete vehicle on each leg of a

request and just compute vehicle flows that consistently support the movement of the requests. This implicitly assumes that a vehicle is always completely unloaded at no cost in zero time upon reaching a vertex. From a solution to the model, it is easy to construct routes for concrete vehicles and to assign request legs to concrete routes.
The authors construct a network as follows. They discretize the time dimension and introduce a vertex for each combination of physical location and time period. The relevant physical locations are vehicle start and end depots, pickup and delivery locations of requests, and TLs. The definition of the arc set is very involved and cannot be detailed here. Basically, there are two types of arc: transport and storage arcs. A transport arc (i, j) links two vertices i and j representing different physical locations and having corresponding time periods t_i and t_j if it is possible that j is reached from i in $t_j - t_i$ time periods. A storage arc (i, i') links two vertices i and i' representing the same physical location l and having corresponding time periods t_i and $t_{i'}$ if it is possible to store a request at l for $t_{i'} - t_i$ time periods. There is at most one arc between any two vertices, which means that several requests and several vehicles can use one and the same arc at the same time. The network is acyclic. The vehicle demands and supplies, which are needed in a network flow model, are given at the start depot vertices of the first and the end depot vertices of the last period respectively and indicate the number of available and required vehicles at the respective locations in the respective period.
Based on this network, the authors present a formulation using general integer arc variables x_{ij}^k indicating the number of vehicles of type k moving from i to j and continuous variables q_{ij}^r indicating the flow of requests. Hence, splitting of load is also allowed. The model has only five types of constraint.
Operation s. is achieved by the network structure, the use of flow conservation constraints for vehicles *and requests*, and the following two types of constraint:

$$\sum_{r \in R} q_{ij}^r \leqq \sum_{k \in F} Q^k x_{ij}^k \quad \forall\ (i,j) \in A^{transport}, \tag{6}$$

where Q^k is the capacity of a vehicle of type k, and

$$\sum_{r \in R} q_{ij}^r \leqq Q_{ij}^s \quad \forall\ (i,j) \in A^{storage}, \tag{7}$$

where Q_{ij}^s is the capacity of storage arc (i, j). (6) require that the overall request flow along a transport arc not exceed the capacity provided by the vehicles moving along this arc, and (7) require that the overall request flow along a storage arc not exceed the storage capacity of the arc. In conjunction with the abovementioned assumption of arbitrarily fast transshipments at zero cost, these constraints also ensure load s.
The paper is a modelling paper, so the authors propose several potential exact and heuristic solution approaches, but do not implement one and do not perform computational experiments.

Mues/Pickl [95] describe two column generation approaches using path-based MIP models for pickup-and-delivery problems with time windows. They consider a limited fleet of heterogeneous capacitated vehicles and do not discretize the time, contrary to Grünert/Sebastian.
In the first model, there is exactly one TL l. The network underlying the first model contains one vertex for the start and one for the end depot of each vehicle, one vertex for each pickup and each delivery of a request, and one pair (v_u^r, v_l^r) of vertices for every request r with pickup and delivery locations r^+ and r^-. v_u^r represents the unloading of r at l from a vehicle, v_l^r the loading, which is done by another vehicle. The network is assumed to be a complete graph. An arc (v_u^r, v_l^r) for any r can only be traversed by r, but not by a vehicle or by any other request. Operation s. is achieved by the following constraint types in the master problem:

$$\sum_{p \in P} a_{rp}^i \lambda_p = 1 \quad \forall\ i \in \{1,2\}, r \in R \tag{8}$$

$$\sum_{p \in P} a^3_{rp} \lambda_p \leqq 0 \quad \forall\, r \in R \tag{9}$$

a^1_{rp} and a^2_{rp} are binary coefficients. a^1_{rp} equals one iff path p performs either request leg $r^+ \to l$ or $r^+ \to r^-$. a^2_{rp} equals one iff p performs either leg $r^+ \to r^-$ or leg $l \to r^-$. a^3_{rp} is the arrival time at l of the vehicle performing path p iff p transports r to l, the negative of the departure time from r^+ or l if p transports r to r^-, and zero otherwise. Due to the construction of the network, request leg sequences can be unequivocally determined from the vehicle paths.
In the second model, several potential TLs are present; hence, each request may be transshipped several times. A network is constructed which, in addition to the vehicle depot and request vertices as in the first model, contains one vertex for every pair (l, k) of potential TL l and vehicle k. By duplicating such vertices, it is possible that a vehicle visits each TL more than once. The network is again assumed to be complete. Only vehicle k can visit a transshipment vertex corresponding to a pair (l, k), and only requests can move along arcs between transshipment vertices corresponding to the same physical location. This time, explicit paths for vehicles and requests are computed. The explicit consideration of request paths guarantees that the requests are correctly transported (without 'jumping' from one vertex to another) and ensures correct timing of the movements of each single request. Furthermore, the request paths are an elegant way to define the load s. at the TLs. Operation s. is achieved by the explicit computation of request paths and, similar to [68], by introducing constraints in the master problem which require, for each arc connecting vertices corresponding to different locations, that the capacity provided by the vehicle using the arc be greater than or equal to the capacity needed by the requests using the arc. Moreover, to ensure temporal operation s., it must be guaranteed that a request r can only be reloaded at a vertex (l_1, k_2) after r was unloaded at a transshipment vertex corresponding to a pair (l_1, k_1). To this end, additional coupling constraints in the master problem are necessary. This leads to similar difficulties for the solution of the pricing problems as those described in [76], see Section 5.4.
The authors sketch potential solution algorithms based on column generation and briefly report on promising computational results for the first model. They do not give details on how to solve the pricing problem in the second model.

Cortés et al. [38] study a PDPTW with transshipments in the context of passenger transport. They develop an MIP formulation based on the following network. There is one vertex for the start and one for the end depot of each vehicle, one vertex for each pickup and each delivery of a request, and two vertices l_u and l_r linked by an arc (l_u, l_r) for each TL l. At l_u (l_r), requests may be unloaded (reloaded). Each vehicle may use each such vertex pair at most once. However, more than one vehicle can visit each pair, so that transfers between more than two vehicles at the same location are possible. To model that a vehicle visit a TL more than once, an arbitrary number of duplicate vertex pairs can be introduced for each TL. The determination of the optimal number of duplicates, though, is not trivial and is not addressed in the paper.
The authors introduce standard arc flow variables x^k_{ij} modelling the flow of vehicles, and continuous time variables t_i, indicating the arrival time at pickup or delivery vertex i, and $t^k_{l_u}$ and $t^k_{l_r}$, indicating the point in time when k arrives at, respectively, leaves TL l. Note that there is no vehicle index on the first time variable type, because each pickup or delivery vertex is visited at most once. Moreover, they use binary z^{kr}_j variables for modelling the flow of requests. z^{kr}_j equals one iff vehicle k carries request r upon reaching vertex j. This means that, contrary to Mues/Pickl, no explicit paths are constructed for the requests; there are no flow variables for the requests.
The authors demonstrate that the requirement $z^{kr}_j \in \{0, 1\}$ will be fulfilled by any feasible solution to the LP relaxation of the model for all vertices j except the transshipment vertices. Consequently, there are fewer binary variables in the model than when flow variables also for the requests would have been used. The formulation contains 24 types of constraint; 4 types are necessary to logically link the x^k_{ij} and the z^{kr}_j variables.

Operation s. is achieved by the z_j^{kr} variables and the following type of constraint:

$$z_{l_u}^{k_1 r} + z_{l_r}^{k_2 r} = 2 \Rightarrow t_{l_u}^{k_1} \leqq t_{l_r}^{k_2} \quad \forall\, l \in L_{tr}, k_1, k_2 \in F, i \in R, \tag{10}$$

where L_{tr} is the set of TLs. (10) ensures that a request r can only be picked up by a vehicle at a TL l if the request has been transported to l before.
In addition, the z_j^{kr} variables ensure load s. since they represent the implicit assumption that only complete requests change vehicle at TLs.
Cortés et al. develop a branch-and-cut algorithm based on Benders decomposition ([13]) and the combinatorial Benders cuts introduced by Codato/Fischetti [35]. This approach avoids the use of the Big-M technique, which is normally used in arc-based formulations of time-constrained VRPs or PDPs and makes for very weak LP relaxations. The authors add symmetry-breaking constraints and additional cuts to tighten their formulation.
The authors compare their method with a direct solution of their formulation by a standard MIP solver on small test instances and report that the former approach saves about 90 % computation time.

Wen et al. [123] consider the vehicle routing problem with cross-docking (VRPCD). In this problem, a set of pickup and delivery requests is given. A set of identical vehicles is stationed at a cross-dock. The vehicles are used to drive one pickup route, picking up goods from suppliers and bringing these goods to the cross-dock, and one delivery route, delivering goods from the cross-dock (after unloading and reloading there) to the suppliers' customers. All routes start and end at the cross-dock. That is, all routes include either only pickup customers or only delivery customers or are empty routes (if a vehicle is not used). The suppliers and customers have time windows associated with them. The problem is similar to the first model studied in [95], where also only one TL is present.
The authors present an MIP model for their problem, based on a network with one vertex for each pickup and each delivery of a request, and four vertices representing the cross-dock, two of which represent the start and end vertices for the pickup routes, and the other two the start and end vertices for the delivery routes. The model uses 9 different types of variable and 20 types of constraint. To ensure a correct course of action at the cross-dock, 7 types of constraint are necessary. Load s. is non-trivial, because the size of a request determines the unloading and reloading time, but it is simplified by the fact that if a vehicle visits the cross-dock after its pickup route, it unloads all requests it will not deliver and loads all requests it will deliver, but has not picked up. Hence, the decision which requests to pick up and which ones to deliver determines load s.
The model is not used for computational purposes. Wen et al. solve their problem with an extension of the unified tabu search (TS) heuristic presented in Cordeau et al. [37], embedded into an adaptive memory procedure (AMP). An initial set of routes is constructed by sequential insertion of pickup or delivery vertices in non-decreasing order of radial angle. The adaptive memory (AM) performs a population management. A constant number of routes is stored in the AM. The routes from the last iteration of the TS are added to the AM with a fitness value corresponding to their duration. A corresponding number of routes with lowest fitness value is removed from the AM. The selection of the routes from the AM for the next iteration of the TS is performed according to a roulette-wheel rule. When a route is selected, all other routes covering a customer covered by the selected route are removed. If unvisited customers remain, they are assigned to empty vehicles by a sweep heuristic. The authors use only the relocate move (move one customer from one route to another), but apply it in two different ways, as an exhaustive and as a non-exhaustive search. The TS alternates between the two and stops when no improved solution was found after a certain number of iterations.
Similar to [42], Wen et al. use two techniques to tackle the interdependence problem: (i) They use the unified TS heuristic, which allows intermediate infeasible solutions and penalizes them in the objective function. (ii) They perform an incomplete evaluation of moves. With respect

to (i), Wen et al. consider the violation of vehicle capacity, route duration and time windows. The objective function is the weighted sum of the overall travel time and these three values, each weighted by dynamic self-adjusting parameters. As regards (ii), the authors skip the time-consuming computation of the overall violation of route duration and time windows, because this would require an update of the schedules of all routes. Instead, they decide whether or not to discard a move based only on the change in travel time and capacity violation. These changes can be computed fast, because these values change only in the two routes directly affected by a relocate move.
In the computational experiments, the authors solve real-world instances with 200 node pairs. They restrict the computation time to five minutes and are still able to solve the instances to within 5 % of a lower bound provided by the sum of the lower bounds for the two independent VRPTWs for the pickup and the delivery customers.

Oertel [100] is a Ph.D. thesis dedicated to pickup-and-delivery problems with transshipments. Splitting of load is not allowed. The paper is a theoretical one in the first place. Several structural results are established and special cases of PDPs with transshipments are examined.
In the second part of his thesis, the author develops a tabu search algorithm for PDPTWs with transshipments. He considers the case with at most one transshipment per request and with two potential TLs. Consequently, there are three possible ways of performing a request: direct transport, one transshipment at the first TL, one transshipment at the second TL.
The first step of the heuristic consists in determining a leg sequence for each request. To do so, a special problem, called k-star hub problem, is solved, which is a special case of a PDP where each request is reloaded at most once and each vehicle route either serves only one request or visits a TL and a pickup or a delivery location. In the first part of his thesis, the author describes this problem in detail and shows that for $k \in \{0, 1\}$, it can be modelled and solved exactly in polynomial time as a network flow problem.
The second step of the heuristic consists in a sequential best insertion heuristic to solve a PDPTW with precedence constraints on the requests, but without transshipments. The requests in this PDPTW, of course, correspond to the legs of the leg sequences determined in the first step for each original request. For example, if request r is to transport some good from location r^+ to location r^-, and if the selected leg sequence for r is $r^+ \to l_1$ and $l_1 \to r^-$, then in the second step, the two requests/legs $r^+ \to l_1$ and $l_1 \to r^-$ have to be performed, and the second one can only be picked up after the first one has been delivered. To ensure this latter condition, the heuristic inserts $l_1 \to r^-$ only after $r^+ \to l_1$ has been inserted, so that it is known when the good to be transported is available for pickup at l_1. Moreover, each time a leg is inserted into an existing route, the temporal feasibility of this and also of all other routes potentially affected by this insertion is checked by performing a 'linear search' (p. 66) to test for violations of time windows.
The third step of the heuristic consists in local search embedded in tabu search. The author uses only one very simple neighbourhood: In each iteration, one original request is removed from the current solution, that is, all legs of the selected leg sequence for the request are removed. As there are only three possible leg sequences for a request, to reinsert the removed request, all three are tried.
In the presence of time windows, the solution of the k-star hub problem computed in the first step may not allow to construct a feasible solution in the second step. This is addressed by allowing infeasible solutions in the second and third step and introducing a weighted penalty term in the objective function for violating time windows and vehicle capacities. The weights are dynamically adjusted in the course of the algorithm.
Computational experiments are performed with randomly generated and with real-world instances of up to 70 requests. The results of the experiments are compared to those obtained by a 'commercial solver' and by a column generation approach, namely, the first one described in Mues/Pickl [95]. The heuristic clearly outperforms the other approaches.

Bock [18] considers a practical, real-time, dynamic pickup-and-delivery problem with time windows, several transshipment possibilities and outsourcing options in the context of multi-modal freight forwarding in Western Europe. Dynamism is introduced by new requests occurring over time and by considering breakdown and deceleration of vehicles, traffic congestion and route blockage. The real-time aspect is accounted for by performing rolling horizon planning. Snapshots of the current situation are either taken time-based (in intervals of between 45 and 180 seconds) or event-based (whenever a new request appears or a dynamic disturbance occurs).
Western European freight forwarding networks consist of local depots and hubs. Each potential customer location is assigned to a local depot. Consequently, four types of leg sequence are possible for a request: (i) Single-leg direct transport, (ii) two legs with one transshipment at the local depot assigned to the pickup location or the local depot assigned to the delivery location or at a regional hub, (iii) three legs with two transshipments at two of the aforementioned locations, or (iv) three legs with two transshipments at two arbitrary TLs (any local depot or hub).
The author presents an exceedingly complex arc-variable based MIP model for his problem. The model contains 14 (!) types of decision variable and 22 types of constraint. Similar to [68] and [95], explicit variables for representing the itineraries of vehicles *and requests* are used. No computational experiments are performed with the formulation.
Bock proposes the following heuristic solution procedure: An initial solution is computed by starting with a route plan routing all available vehicles directly from their start to their end depot. Requests are then inserted iteratively one by one. In each iteration, one of the requests causing the smallest increase in the overall driving time of the current route plan is inserted. The insertion is done by examining, for each of the four different types of request leg sequences described above and each resulting leg, different combinations of vehicles, external carriers and hubs. The interdependence problem is then dealt with in the following way: After inserting a leg into a vehicle route, the schedule of the complete route plan is updated, storing all routes that are affected by a change in another route on a stack. At first, only the route into which the new leg was inserted is on the stack. This route is removed from the stack, its schedule is updated, and when a loading or unloading activity is encountered along this route, the routes affected by these activities are pushed onto the stack etc. This is repeated until the stack is empty or an infeasibility is detected. After examining all four potential types of leg sequences for the request to be inserted, the leg sequence with minimal costs is kept for possible implementation.
To improve a given solution, the following four-stage local search procedure is used: In stage $s \in \{1, \ldots, 4\}$, s requests are selected randomly and removed from the current plan. Each request is then re-inserted by the procedure just described. If the re-insertion of the s requests leads to a reduction of overall costs, the new solution is used and another s requests are chosen, otherwise, the existing route plan is kept and s is increased. This means that only improving solutions are considered. Preliminary tests showed that this significantly outperformed approaches allowing temporary deteriorations.
The author performs computational tests with several scenarios regarding the degree of dynamism and the use of time- or event-based re-planning. An approach with time-based re-planning every 90 seconds turns out best. The tests are performed with instances comprised of up to 372 requests, 55 vehicles and 399 potential TLs.

Jacobsen/Madsen [80] are the first to describe an application of a two-echelon vehicle routing problem. The context is newspaper distribution. In their problem, there is one level-0 location, a central printing facility, from which newspapers have to be delivered to uncapacitated TLs by large support vehicles. At the TLs, small task vehicles are used to distribute the newspapers to retailers. Each retailer location is a potential TL. There are time windows at the retailers, route duration constraints for the routes of the second stage, and production rate constraints for the routes of the first stage: Printing starts at 6:15 h, lasts until 8:45 h, and 1,000 newspapers are printed per minute. All deliveries must be finished by 11:30 h.

The authors present three heuristics for the problem. The first one constructs a spanning tree whose root is the printing facility, and where each retailer with more than one successor is interpreted as a TL. Initially, the tree consists only of the printing facility. The retailers are first sorted by non-decreasing time difference between their respective latest delivery time and the direct travel time from the printing facility. Then one retailer at a time is connected to the tree by an appropriate arc which should not exceed a certain travel time and should not be incident to certain vertices of the tree. The idea behind this is that such a spanning tree represents a route plan from which the last arc of each route (from the last TL back to the printing facility or from the last retailer back to the TL where the route started) is omitted. Operation s. is implicit and results from the structure of the constructed arborescence.
The second one is a three-stage heuristic. The first stage consists in a location-allocation procedure, where retailers are allocated to a predefined number of TLs, taking into account the 11:30 h delivery deadline. The second and third stages determine routes for the task and support vehicles respectively by means of the savings heuristic, based on the results of the first stage, taking into account customer time windows and newspaper availability.
The third heuristic is also a three-stage procedure. First, routes for task vehicles are constructed by the savings heuristic with the printing facility as the unique depot. Second, a drop heuristic is used to locate TLs, taking into account the retailer time windows. Initially, a TL is located at the first retailer of each route. Third, routes for the support vehicles are determined as in the second heuristic.
It is very interesting to note that despite the considerable size of the problem (4,510 customers), the authors report storage requirements for the solution procedures of between 164 and 380 kilobytes of main memory only. The code for each of the heuristics is stated to contain no more than about 500 FORTRAN statements.

Lin [88] considers a PDPTW with homogeneous, autonomous, uncapacitated vehicles. Documents have to be picked up from customers and delivered to a central depot. There is a pickup as well as a delivery time window.
The author allows transshipments as an improvement option, similar to [108]. Given a set of feasible routes, a vehicle is allowed to travel to the last location on another vehicle's route and transfer load. For two given routes r_1 and r_2, there is at most one load transfer operation (either from r_1 to r_2 or vice versa). In either case, the location where the transfer takes place is fixed (the last pickup location of either r_1 or r_2), and the amount of load transferred is also fixed (the complete collected load of either r_1 or r_2). This avoids delays on the second vehicle's route, and hence mitigates the interdependence problem, but of course decreases the additional degrees of freedom gained by considering transshipments. Load s. is trivial, and operation s. is eased considerably, because the passive vehicle acts as lead vehicle and determines the location and the time of the load transfer: It is simply checked whether it is possible to go from the last pickup in r_1 to the last pickup in r_2 before the end of the latter's dynamic time window (assuming that r_1 transfers its load to r_2), and whether it is possible to arrive at the depot before the latest delivery time of r_1 and r_2 when leaving r_2 at the latest possible time.
The author gives an MIP model based on the following network: There is one vertex d for the central depot. For each feasible route r, there are two vertices i_r and j_r. i_r denotes the fact that route r is performed, and j_r denotes the last pickup in r. If a vehicle performs route r_1 and does not transfer its load, this is represented by the route $d \to i_{r_1} \to j_{r_1} \to d$. If a vehicle performs route r_1 and transfers its load at the last pickup location of route r_2, this is represented by the route $d \to i_{r_1} \to j_{r_1} \to j_{r_2} \to d$. The MIP model is straightforward and has only one type of variable, a binary arc variable specifying whether or not an arc is used, and five types of constraint. Operation s. with precedences is ensured by the network structure, so that no time variables are necessary.
To construct the routes for the above network, the author proposes two tree search procedures which essentially perform complete explicit enumeration. Interestingly enough, the com-

putational experiments show that the procedures work for instances with 100 requests (most probably due to tight time windows). The author compares the results obtained with the tree search procedures (that is, solutions to the problem without transshipments), the constructive heuristic by Lu/Dessouky [90] (which also does not consider transshipments), and the solution of the MIP model by a standard solver. The best solution quality is obtained when multiple use of vehicles and load transfers are allowed; however, this also incurs the highest computation times. It turns out that allowing load transfers considerably reduces the number of vehicles, whereas the savings in total distance are rather small.

Aldaihani/Dessouky [2] study the problem of integrating on-demand dial-a-ride passenger transport services (taxis) and fixed-schedule public transport lines (buses). Essentially, this is a PDPTW where some vehicles perform fixed routes. The authors assume that each request (person) can either be transported directly from an origin to a destination by a taxi or be transported to a bus stop close to the origin by a taxi, travel by bus to a bus stop close to the destination, and be transported by another taxi to the destination. The number of possible request leg sequences is in general much larger than two, because it is not specified which bus stops should be used. This has to be determined as part of the problem.
The authors develop a three-stage heuristic procedure. In the first stage, for each request, a set of possible leg sequences is determined. In the second stage, an initial feasible solution is computed using the results of the first stage, that is, a concrete leg sequence for each request is selected, and routes for the taxis are computed. This is done by an insertion procedure. In the third stage, the solution from the second stage is improved.
In stage one, to determine leg sequences for a request r from origin i to destination j and a distance from i to j of d_{ij}^{direct}, the authors proceed as follows. For a possible leg sequence $i \to b_1 \to b_2 \to j$ with partial distances d_{ib_1}, $d_{b_1b_2}$, and d_{b_2j} and overall distance $d_{ij}^{indirect} = d_{ib_1} + d_{b_1b_2} + d_{b_2j}$, three criteria are used: (i) $(d_{ib_1} + d_{b_2j})/d_{b_1b_2}$ must be greater than or equal to a threshold, (ii) $d_{ij}^{direct}/d_{ij}^{indirect}$ must be less than or equal to a threshold, and (iii) d_{ij}^{direct} must be greater than or equal to a threshold. The authors do not specify which possible leg sequences they check in this stage. For each request, the sequence with minimal $d_{ib_1} + d_{b_2j}$ value is selected.
In stage two, all selected direct legs and first legs $i \to b_1$ of three-leg sequences are sorted by non-decreasing earliest pickup time and inserted into existing taxi routes, taking into account capacity and time window constraints. New vehicles are added if the already used vehicles cannot accommodate a request. Each request is inserted into the route whose overall distance increases the least due to the insertion. If a first leg of a three-leg sequence has been inserted, the corresponding third leg is added to the set of leg sequences to be inserted. The pickup time window for the second leg is determined according to the given bus schedule. This is repeated until all requests are planned.
In stage three, for each three-leg sequence in order of non-decreasing earliest pickup time, all legs are removed from the solution, and all other leg sequences computed in stage one for the respective request are checked for whether their insertion leads to an improvement of the overall solution in terms of overall distance travelled by the taxis. The best sequence is selected. That is, once the best leg sequence for a particular request is found, this sequence constitutes a fixed part of the final solution.
After that, tabu search is used to improve the solution found in stage three using an intra-route and an inter-route neighbourhood. In the intra-route neighbourhood, stops are relocated within their assigned route. In the inter-route neighbourhood, stops are moved from one route to another. Only feasible moves are considered: Moves must maintain the precedence constraint (pickup before delivery, only for direct legs), the vehicle capacity, and the delivery time windows. The latter requirement considers the interdependence between legs $i \to b_1$ and $b_2 \to j$ for three-leg sequences. However, it remains unclear how the authors maintain feasibility of a route plan in view of the fact that the insertion of a request stop into a route may delay the pickup of another request r on the same route by so much that the route which is to perform the third

leg of r becomes infeasible. Ensuring this is non-trivial and time-consuming. One way how to do this is described in Bock [18].
The authors perform computational tests on real-world data with up to 155 requests and 12 vehicles and obtain promising results.

Shang/Cuff [116] and **Thangiah et al. [120]** consider a pickup-and-delivery problem with time windows in the context of parcel delivery. Several requests may have to be picked up from or delivered to the same locations during different time intervals. This means that a location may be visited by several vehicles. There are only autonomous, uncapacitated vehicles. Transshipments are allowed at any location between any two vehicles. Time is discretized into 15-minute intervals. The authors use an insertion heuristic to compute solutions. If a request $r^+ \to r^-$ can be picked up but not delivered in time by a vehicle k, it is checked which locations k visits after r^+, and which other vehicles visit one of these locations after k. If one of these vehicles visits r^- afterwards, this vehicle can deliver the request. This simple principle ensures both operation and load s.

Mitrović-Minić/Laporte [93] study a pickup-and-delivery problem with time windows and transshipment possibilities in the context of parcel delivery with homogeneous, uncapacitated vehicles. A request is allowed to be transshipped at most once at one of several potential TLs.
The authors develop a two-phase heuristic. In the first phase, an initial solution is computed by a multi-start cheapest insertion procedure, which is started with different random initial orderings of the requests. In the second phase, a given solution is improved by a descent procedure which successively removes and re-inserts each request. The incumbent solution is updated upon improvement and the procedure stops if an iteration does not improve the best solution.
The heuristic is based on the idea of determining request legs: For each request, there are $L+1$ different leg sequences, where L is the number of potential TLs. Each time a request is to be inserted, each of the $L+1$ leg sequences is evaluated. For the single-leg sequence as well as for each leg of each two-leg sequence, the best positions for inserting are determined, and the cheapest among the $L+1$ sequences is inserted.
To ensure operation s. between the vehicles performing a request, it must be guaranteed that any second leg from the TL to the delivery location is picked up at the TL only after the first leg has been delivered. This is achieved in the following way: Let $r^+ \to l, l \to r^-$ be a leg sequence. First, $r^+ \to l$ is first inserted into a route. This results in an earliest point in time for the delivery of the load at l. This point in time then becomes the earliest point in time for the pickup of the request at l, that is, for the pickup of leg $l \to r^-$, and the latter is inserted into a route, taking into account this earliest pickup time. Second, $l \to r^-$ is first inserted into a route. This results in a latest point in time for the delivery of the load at l. This point in time becomes the latest point in time for the delivery of the request at l, that is, for the delivery of leg $r^+ \to l$, and $r^+ \to l$ is inserted into a route, taking into account this latest delivery time. The cheaper of the two alternatives is selected if feasible.
However, similar to [2], also Mitrović-Minić/Laporte do not elaborate on how feasibility of a route plan is maintained in view of the fact that the insertion of a request (leg) into a route may delay the pickup of another request r on the same route by so much that the route which is to perform a potential second leg of r becomes infeasible.
The authors perform computational experiments with random test instances containing up to 100 requests and 4 TLs. The results show significant savings through the use of TLs compared with the situation without transshipments. The savings increase with the number of requests and the length of the time windows and are more significant for instances with clustered request locations.

Gørtz et al. [67] is a theoretical paper where complexity results for dial-a-ride problems with transshipments requiring operation s. with precedences are derived and a route-first-cluster

second $\mathcal{O}(\log^2 |L| \log |R|)$ approximation algorithm is proved, where $|L|$ is the number of pickup and delivery locations and $|R|$ the number of requests. Each location is a potential TL. No computational experiments are performed.

Fügenschuh [56], [57] considers the problem of synchronizing morning start times of schools and school bus trips (sequences of bus stops visited by the same bus). Several trips may visit the same school.
Fügenschuh models this as a VRPTW. Bus trips are interpreted as tasks/customers/vertices, and an arc linking two tasks i and j corresponds to the deadhead trip between the last stop of trip i and the first stop of trip j. The objective is to minimize the number of buses used.
An MIP model is developed using binary x_{ij} and continuous time variables t_i and t_s indicating the start time of trip i and school s respectively. Considering the driving time t_{is}^{drive} from the first stop of trip i to the school s and a minimal and a maximal waiting time $t_s^{wait,min}$ and $t_s^{wait,max}$ for the pupils after the arrival at their respective school and the beginning of the first lesson, the following constraint type ensures operation s. with precedences at different locations:

$$t_i + t_{is}^{drive} + t_s^{wait,min} \leqq t_s \leqq t_i + t_{is}^{drive} + t_s^{wait,max} \tag{11}$$

Note that there are no vertices for the schools. As stated above, all vertices correspond to bus trips. (School vertices could easily be introduced, but the interesting thing is that the model works without them.)
[56] presents a greedy heuristic followed by local search, [57] a branch-and-cut algorithm. Both procedures are successfully tested with real-world instances.
In each iteration of the greedy heuristic, an arc (i, j), that is, a deadhead trip from the final destination of trip i to the start location of trip j, is added to the solution. The best arc with respect to a scoring function is selected. The scoring function is the weighted sum of (i) the length of the deadhead trip, (ii) the temporal difference between the trips, and (iii) the flexibility for concatenating the trips. Suitable weights are found with a descent heuristic called improving hit-and-run. After each iteration, the time windows of all vertices are updated, taking into account the last added arc. Due to the interdependence problem, this time window update is necessary to detect infeasibilities resulting from the addition of the last arc. After the greedy heuristic, a local search is performed on the solution found. The author notes that local search is computationally difficult, because the evaluation of a move is so costly due to the interdependence problem. He therefore uses only one neighbourhood which essentially performs a two-opt arc exchange. After that, concrete values are assigned to the trip and school start times. This is done by sequentially setting one start time after the other to a concrete value within the remaining time window of the respective school as obtained at the end of the greedy heuristic. Also in this final step, after fixing one time variable, the bounds for the other time variables have to be updated.

Concluding remarks on operation synchronization with precedences

The transition from exact operation s. to operation s. with precedences is a gradual one. Many approaches presented in this section could be extended to consider exact operation s., and also some approaches for exact operation s. could be modified for operation s. with precedences.
The survey on operation s. shows that interdependencies between vehicles induced by the temporal aspect of operation s. (with precedences as well as exact) are tedious to handle and, above all, to program in heuristics: It is computationally costly to maintain feasible solutions during the complete solution process, and the influence of a move on the objective function cannot be computed efficiently.

5.6 Movement synchronization at the depot

In this section, problems are discussed where two types of non-autonomous object, motor vehicles and drivers, perform movement s. at a central depot; that is, the objects perform a complete route together. The following table gives an overview.

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[34]	Food distribution	Lorries and drivers	Movement at depot	–	Heuristic: routing via seed points, assignment of vehicles and drivers afterwards
[106]	Propane delivery	Lorries and drivers	Movement at depot	–	Heuristic: Construction by greedy best insertion, improvement by hybrid LNS/tabu search
[85]	Limousine rental	Limousines and drivers	Movement at depot	–	Constraint programming followed by local search using simulated annealing
[125]	Dial-a-ride	Vehicles and drivers	Movement at depot	Path	Four-stage heuristic with local search, column generation
[126]	Postal delivery	Vehicles and drivers	Movement at depot	Arc	Two-stage heuristic: VRPTW, assignment problem
[107]	Household activities	Cars, drivers and passengers	Movement at depot	Arc	MIP-based heuristic

Table 4: Papers on movement synchronization at the depot

Chung/Norback [34] describe a cluster-first-route-second heuristic to solve a practical vehicle routing problem in the food distribution sector. To solve the problem, the authors first create clusters via seed points, similar to the Fisher-Jaikumar principle. Additional customers are added to clusters by an insertion heuristic. To consider vehicle capacities and driver working times, the respective greatest capacities and longest working times of the still available vehicles and drivers are checked. If the insertion of a customer into a route leads to the violation of one of these two constraints, the customer is not inserted and the route is assigned a suitable vehicle and a suitable driver. Essentially, this means that at first, routes for the larger vehicles are planned. Nevertheless, routes are initially not planned for concrete vehicles, but routes are considered objects which fulfil tasks and which are assigned resources, that is, vehicles and drivers.

Prescott-Gagnon et al. [106] present a hybrid large neighbourhood search (LNS)/tabu search heuristic for the planning of propane delivery to customers. Their procedure is able to plan drivers and vehicles independently, considers multiple replenishment possibilities for the vehicles at different fillup stations, a maximum shift time for the drivers, and mandatory as well as optional customers. It is assumed that a driver may only perform one shift and that the service times at the customer and fillup vertices are known and fixed, i.e., independent of the customer demand and the vehicle load. The authors use a greedy construction heuristic which sequentially builds up routes for driver-vehicle pairs by inserting the temporally closest customer (with respect to travel time from the current end of the route and customer time window). In the LNS algorithm, four different heuristics are used to determine the nodes which are fixed for the tabu search heuristic that constitutes the reconstruction phase. These fixing or (depending on the perspective) removal operators are similar to the ones presented in, e.g., Ropke [109]: random removal, proximity operator, time operator and longest detour operator. In the tabu search phase, the following move types are used: (i) Exchange moves remove one customer from a route and insert it into another one. (ii) Station insert/remove moves either remove a fillup station from a route, replace a removed station by another one, or insert a new station. (iii) Driver switch moves try to switch a pair of drivers, i.e., have one driver drive the other driver's route and vice versa. As is usual in tabu search, a best move search is performed, and the best non-tabu move found is performed. Infeasible solutions are allowed in the course of the algorithm, and

such solutions are assigned a penalty cost. The penalty cost is dynamically adjusted to increase or decrease the number of infeasible solutions visited.

Laurent/Hao [85] consider the problem of simultaneously scheduling vehicles and drivers for a limousine rental company in the Paris region. Each limousine can perform only one request at a time, but may perform several requests before returning to the depot and may drive several routes. The same holds for the drivers. To fulfil a request, one limousine and one driver are needed. The problem is highly dynamic, as requests for limousines may arrive at any time. This aspect is however ignored algorithmically; the authors only set tight running time limits for their solution procedure.
The objective function comprises several criteria. The most important one is the minimization of the duration of the unfulfilled requests. (The overall duration of a request determines the revenue generated by the request. It is possible not to fulfil a request due to lack of resources.) Further objectives, among others, are the minimization of the number of vehicles and drivers used and the minimization of the overall waiting time of drivers between two successive requests. There are several side constraints, among them the seat capacity of a vehicle, special skills the drivers need to have to serve a request, and a maximal duration of a driver shift.
To model the problem, the authors use a constraint programming approach where the set of requests is considered as the set of decision variables. Each variable has an associated domain consisting of vehicle-driver pairs. The solution approach consists of two phases. First, an initial feasible solution is constructed by means of a greedy heuristic similar to the well-known best fit decreasing strategy for the bin packing problem, using constraint programming techniques for domain reduction. Second, an improvement procedure based on local search embedded in a simulated annealing metaheuristic is performed, using the following neighbourhood: First, a variable x is randomly selected. Then, it is assigned a new value, that is, a new vehicle-driver pair, and the old value is stored. Third, if assigning the value does not make the overall solution infeasible by violating a constraint, the new assignment is accepted. Otherwise, all other variables which now have inconsistent values are assigned a new value such that the overall solution remains feasible. (A variable y may have an inconsistent value if it is assigned a vehicle-driver pair that after the change of the assignment of x is no longer able to fulfil the request corresponding to y.) This may mean that some of the variables become unassigned, that is, the corresponding requests are no longer fulfilled. Fourth and finally, all unassigned variables are scanned for whether they can be re-assigned their previously stored value. The term 'simultaneous' in the paper title refers to the fact that the neighbourhood consists in the simultaneous exchange of a driver and a vehicle.
The computational experiments show that the developed algorithm consistently and considerably outperforms existing manual plans. The algorithm has already been included in the software system of the rental company.
The approach taken in this paper is in sharp contrast to the one taken by De Rosa et al. [42], Wen et al. [123], and Prescott-Gagnon et al. [106]: Whereas Laurent/Hao maintain feasibility at a large computational cost, considering the complexity of the chosen neighbourhood, the other authors explicitly allow infeasible solutions as a means of coping with the complex constraints of their problems. Also, Laurent/Hao point out that the use of powerful neighbourhoods is decisive to obtain good performance of local search algorithms. This is in contrast to the findings of Wen et al.

Xiang et al. [125] consider a static dial-a-ride problem with several complicated constraints. In particular, the problem requires the assignment of a vehicle and a driver for each route. Both drivers and vehicles can perform more than one route, and on each route, a driver can be assigned a different vehicle and vice versa. There are different types of customer, and each vehicle has a limited number of seats for each customer type. For the drivers, there are restrictions on the duration of one route and on the overall working time per day and restrictions on the

minimum time between two consecutive routes a driver drives. Moreover, there are vehicle-driver compatibility constraints.
Essentially, the authors solve the problem by a four-phase heuristic procedure. The details of the overall heuristic are really very sophisticated and cannot be presented in detail here. Basically, it works as follows. The first phase consists of preprocessing steps to speed up the subsequent computations. In the second phase, a feasible initial route plan is constructed by (i) determining subsets of customers that cannot be on the same route due to time window restrictions, (ii) constructing a list customers sorted in non-decreasing order of the end of their delivery time windows, and (iii) computing routes via a procedure similar to the sweep algorithm based on this list. In the initial plan, no concrete vehicles or drivers are assigned to the generated routes, but rather only vehicle and driver types (there are three such in the test instances the authors use for their computational experiments). In the third phase, the initial solution constructed in the first phase is improved by an intensification step (local search with four different neighbourhoods) and a diversification step (using a kind of penalty function). Only in the fourth phase is movement s. ensured by assigning concrete vehicles and drivers to routes. This is done by assigning to a given scheduled route a vehicle/driver whose longest contiguous free time within the overall planning horizon, after inserting the route into the vehicle's/driver's schedule, is maximal over all vehicles/drivers.
Additionally, Xiang et al. develop a column generation approach for their problem. For each feasible route p *with assigned vehicle and driver*, there is one variable λ_p. In addition to the covering and convexity constraints, the master problem contains one constraint of the form

$$\sum_{p \in P} a_p^k \lambda_p \leqq n^k, \tag{12}$$

where a_p^k equals one if k performs route p and zero otherwise, for each type k of vehicle or driver of which there is only a limited number n^k available. There is one subproblem for each combination of vehicle and driver type. The subproblems remain benign and can be solved by the standard dynamic-programming based labelling algorithm; no vertex costs arise as in [76], [12], or [52].

Zäpfel/Bögl [126] consider a problem of simultaneous vehicle and driver routing and scheduling, taking into account social legislation on driver working and driving times as well as the option of outsourcing tasks (vehicle routes) to external carriers. The authors examine the situation in local letter mail distribution. Pickup routes and delivery routes (but no combined pickup-and-delivery routes) have to be planned during one week. The pickup routes transport outbound shipments from local post offices to a letter mail distribution centre. Conversely, the delivery routes transport shipments from the distribution centre to the post offices. Each local office must be visited by one pickup and one delivery route per day. The drivers and vehicles may be used for more than one route in the planning horizon. Schedules are planned for both drivers and vehicles, and the authors distinguish between own and external drivers and vehicles. The objective is to minimize the sum of distance-dependent costs for the own vehicles, time-dependent costs for external drivers and vehicles, and overtime costs for own drivers.
The authors set up an ingenious arc-variable based MIP model using 10 types of variable, one of them indexed six-fold, and 18 types of constraint, taking into account all of the above aspects. They index different types of variable over all routes and all vehicle types. 'All routes' here does not mean all possible permutations of all subsets of the set of post offices, as in column generation approaches, but rather means the minimum number of routes that have to be driven with a given fleet. Thus, symmetries are introduced, as it does not make a difference whether a vehicle of type k' is used on the first route and a vehicle of type k'' on the second route or vice versa. This approach, however, facilitates the consideration of concrete own vehicles (which is important for modelling the driver rules), of external drivers and external driver-vehicle combinations. The update of the time variables for a route is basically as usual, but the consideration of driver rules

makes this update extremely complicated. Movement s. is essentially achieved by a constraint type ensuring that, for each route in each period, it is possible to (i) allocate a company driver from the pool of drivers with the company vehicle pool, (ii) assign a leasing driver to a vehicle from the company pool, or (iii) outsource to a company providing drivers and vehicles. This is sufficient: Since vehicles and drivers are re-grouped only between routes, at the central depot, the routing of a driver is completely determined by the route(s) he is assigned to. So, the time variables for the routes determine when a driver returns to the depot and when he can start his next route. Overall, the MIP model is very involved, but no computational experiments are performed with it.
Instead, Zäpfel/Bögl solve their problem heuristically, by decomposing it into a VRPTW and a 'personnel assignment problem' (PAP). First, a feasible solution for the VRPTW is computed, using the available vehicles, via a modification of the I1 heuristic by Solomon [118]. The heuristic is modified to consider several aspects of driver rules. The PAP then tries to find a feasible driver assignment for the VRPTW solution. This is achieved by computing an assignment matrix storing, for each pair (own driver, route), the costs resulting from the driver performing the route and taking those driver rules into account that were not considered in the VRPTW solution step. Then, a complete assignment is computed, using three different strategies, among them a greedy and a random procedure, and taking into account the possibility of using external drivers or external driver-vehicle combinations. After that, an improvement procedure embedded into a metaheuristic follows. The improvement procedure again solves the two partial problems VRPTW and PAP sequentially. For the VRPTW, four standard local search neighbourhoods are employed; for the PAP, the same three heuristics as in the construction phase are used. As metaheuristics, tabu search and genetic algorithms are implemented.

Recker [107] develops a tremendously complex MIP model for the household activity pattern problem (HAPP). In this problem, persons perform a specified set of tasks or out-of-home activities. There are no time windows for the tasks, but for the departure from home and the return. To reach the locations where the tasks are to be performed, vehicles (cars) are available. Multiple departures by a vehicle as well as a person are possible. It is allowed that persons use different vehicles on different routes and that two persons share a vehicle on one route (ridesharing). Hence, the HAPP is a problem of movement synchronization of three non-autonomous objects at the depot.
The model is based on a network with one depot vertex corresponding to the home location, where all routes start and end, and two vertices i^+, i^- for each task. i^+ corresponds to the physical location where the task is to be performed, and i^- represents the delivery and corresponds to the home location. This means that all tasks have the same physical delivery location. To consider ridesharing, there are vertices for dropping off and for picking up household members at the task locations. There are arcs from the dropoff vertex $i^{dropoff}$ of a task i to the task vertex i^+ and from i^+ to the pickup vertex i^{pickup}. For a given task i, a vehicle with at least one passenger is then allowed to either move from a different location to i^+ and from there to another location. Alternatively, if it carries more than one passenger, it can move from another location to $i^{dropoff}$, drop off a passenger, and move on to another location. The passenger can then move to i^+, perform the task, move to i^{pickup} and wait for the vehicle to return and pick him up.
The model uses binary x_{ij}^k arc variables for each household member and the driver seat and the passenger seat of each vehicle. x_{ij}^k equals one iff person or driver seat or passenger seat k travels directly from vertex i to vertex j. Moreover, continuous variables t_i indicating the arrival time at vertex i and time variables for the departure from and the return to the depot for each household member and each driver and passenger seat are used.
The formulation for the HAPP has 49 types of constraint, which is a record. Essentially, movement s. between household members and driver and passenger seats is ensured by constraint types requiring the following:

(i) The number of departures of all household members from the depot vertex or a delivery vertex equals the number of departures of all vehicle seats from the depot or a delivery vertex. (Remember that the physical location of each delivery vertex is the home location.)
(ii) The departure time from and the return time to the depot must be identical for the driver seat and the passenger seat of a vehicle.
(iii) The departure time of a seat from the depot or from a delivery vertex equals the corresponding departure time of a household member.
(iv) At most one vehicle may visit a task, dropoff, or pickup vertex.
(v) The number of persons travelling along any arc (i, j) is equal to the number of seats *that are used* on the vehicle travelling along (i, j).

In addition, two very intricate types of constraint ensure that if a passenger seat visits a task vertex, the corresponding driver seat must visit the corresponding dropoff and pickup vertices. Together with standard flow conservation constraints, these two types of constraint link the itineraries of the driver and the passenger seat of each vehicle. Moreover, in this way it is also ensured that tasks can only be performed by a passenger if this passenger is transported to and from the respective dropoff and pickup vertices

There are no pure passenger transfer locations in the formulation. Moreover, it is not considered that one driver with one vehicle may drop off a passenger at a location and a different driver and/or vehicle picks up the passenger later. Both extensions could easily be introduced though. The author solves small instances of the model heuristically by first solving a version of the problem without ridesharing with a standard MIP solver. Based on this solution, he heuristically generates feasible solutions with ridesharing.

Concluding remarks on movement synchronization at the depot

Although the objects to be synchronized in all papers described in this section are vehicles and drivers, the concrete application contexts are all different. As regards solution approaches, some papers ([34], [125], [126]) exploit the fact that drivers and vehicles may join and separate only at one location by performing, in principle, a decomposition of the problem by object type. Others ([85], [106]) take the opposite way and compute routes for predetermined driver-vehicle pairs.

5.7 Movement synchronization en route

The final type of s. considered is movement s. en route. Papers are discussed where composite autonomous objects consisting of two or more types of elementary autonomous and/or non-autonomous object are required to fulfil tasks. The elementary objects may join and separate at many different locations. Compared to the problems described in the previous section, this adds an additional degree of freedom and, hence, complexity. The subsequent table gives an overview of the problems considered in this section.

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[83]	Staff scheduling	Vehicles and persons	Movement en route, operation with precedences	Arc	Standard MIP solver, greedy heuristic
[73]	Mail delivery	Vehicles and drivers	Movement en route, operation with precedences	Path	Heuristic column generation
[52]	Raw milk collection	Lorries/tractors and trailers/semi-trailers	Movement en route, exact operation, continuously split load	Arc, turn, and path variables	Branch-and-cut, Branch-and-price
[14]	Long-distance road transport	Lorries and drivers	Movement en route	Arc	Standard MIP solver

(continued on next page)

(continued from previous page)

Paper	Application(s)	Objects to synchronize	Types of s.	MIP variable type	Solution approach(es)
[24]	Long-distance road transport	Drivers, lorries with loading capacity, lorries without loading capacity, tractors, trailers, semi-trailers, chassis, and swap-bodies	Movement en route, exact operation, fixed load	–	Holonic multi-agent system
[30]	Seaport container drayage	Tractors, drivers, semi-trailers	Movement en route, exact operation, fixed load	–	Attribute-decision model

Table 5: Papers on movement synchronization en route

Kim et al. [83] study a combined vehicle routing and staff scheduling problem. In this problem, a sequence of tasks has to be performed at customers. That is, at each customer, a certain number of tasks has to be fulfilled in a fixed sequence. A task can begin as soon as the previous task is completed. To fulfil the tasks, different teams of workers are available, all stationed at one central depot. Each team is qualified to perform one specific type of task. The innovative aspect of the problem is that the teams cannot move by themselves between customer locations and the depot. Instead, a set of vehicles to transport the teams is available at the depot. There is no fixed assignment of a vehicle to a team; each team can be transported by any vehicle. A vehicle can transport at most one team at a time. The problem consists in finding a set of routes for teams as well as vehicles, such that all tasks are fulfilled in the correct sequence.
The authors present an MIP model based on a network with one vertex for the depot and one vertex for each customer. This means that customer vertices may be visited more than once by more than one vehicle and more than one team. The model contains 9 types of variable and 20 types of constraint. Movement s. is achieved by binary x_{ijkm} variables which equal one iff the path from i to j is used for transporting team m by vehicle k, and two types of constraint using these variables to ensure the temporal synchronization of the movements of vehicles and teams. Operation s. is ensured by (i) a constraint type requiring that the tasks to be performed at each customer be performed in the correct sequence, (ii) a constraint type stating that the start time of a task at a customer cannot be earlier than the arrival time of the team assigned to the task, (iii) two constraint types linking the arrival and ready times of teams to the arrival times of the vehicles transporting them, and, finally, (iv) three constraint types ensuring the continuity of the vehicle movements.
To solve the problem, the authors develop an astonishingly simple procedure, called dispatching based heuristic algorithm (DBHA), which works as follows. The vehicles, the teams, and the next tasks for each customer are stored in three lists, along with the following information: For the vehicles, their respective locations and ready time for the next transport are stored. For the teams, their locations and the available times for transport to the next location are stored, and for the tasks, the customer number, the number of the task in the respective sequence, and the completion time of the previous task are stored. In each iteration, the procedure selects a triplet (vehicle, team, task) from the lists. Then, the information in the lists is updated to reflect the situation resulting when the selected vehicle transports the selected team to the location of the selected task. The selection is performed on a best-fit basis: For each combination of vehicle, task, and available suitable team, a fitness criterion is evaluated, and the combination with the best fitness value is selected. The fitness value is the weighted sum of six values, each of which represents a characteristic of a combination. For example, one value indicates for how long an available team has to wait for the vehicle. The best-fit selection step is repeated until the task list is empty.
Kim et al. propose a particle swarm optimization algorithm (PSO) for the optimization of the six weight parameters. In this algorithm, 100 problem instances are used as particles. In each

iteration of the PSO, the DBHA heuristic is used to solve each of the 100 instances. The weight parameters are then updated using the obtained solutions.
The authors perform computational experiments with newly set up test instances created from several VRP benchmark instances. The number of customers ranges from 100 to 480, the number of teams and vehicles from 3 to 8. Solutions are computed in less than 20 minutes of CPU time. Overall, the problem and its solution show how simple problems with operation and movement s. can be. The fact that the tasks to perform at each customer have only precedence relationships, but no time windows, makes operation as well as movement s. much easier. In particular, the feasibility of a solution can be ensured very easily. The DBHA procedure skilfully exploits this fact.

Hollis et al. [73] describe a simultaneous vehicle and crew routing and scheduling application for urban letter mail distribution for Australia Post. The problem is a pickup-and-delivery problem with time windows within a 24-hour planning horizon, multiple depots, a heterogeneous fleet of vehicles, and a heterogeneous 'fleet' of drivers. There are compatibility constraints between requests (letter mail), vehicles and drivers. Requests may originate and end at depots, and all drivers and vehicles have a home depot from which they start their schedules and to which they must return at the end of the planning horizon. The task is to determine minimum fixed and variable cost routes and schedules for the vehicles and the drivers such that all requests are delivered to their destinations in time. There is no fixed assignment of vehicles to drivers, that is, a driver may change vehicles during his shift (only at depots), respectively, a vehicle may be driven by several drivers within the planning horizon. The original aspect of the problem is that there are multiple depots where vehicles and drivers may be stationed.
For the solution of the problem, the authors distinguish between the three objects vehicle route, crew schedule, and vehicle schedule. A vehicle route is a sequence of pickups and deliveries which starts and ends at a vehicle depot and which has a concrete vehicle *type* assigned (not a concrete vehicle, unless there is only one vehicle of a particular type). A crew (vehicle) schedule is a schedule for a set of activities to be performed by one driver (vehicle). Based on these three types of object, the authors take a two-phase approach. In the first phase, they determine vehicle routes by solving a pickup-and-delivery problem with time windows, multiple depots and a heterogeneous fleet. In order to be able to determine feasible driver schedules in the second phase (see below), the first-phase problem considers several working time restrictions. The authors develop a path-based MIP model for this first-phase problem, where the decision variables correspond to feasible vehicle routes, and solve the model by heuristic column generation. The column generation procedure is based on enumeration of all one- and two-request routes to get an initial pool of columns. Then, using the dual prices from the restricted master problem, several local search steps, such as including a new request into a route, are performed in each column generation iteration. When no more columns with negative reduced cost can be found or a tailing-off effect is observed, column generation is terminated, and an integer solution is determined by heuristic branch-and-bound.
In the second phase, vehicle and crew schedules are determined taking an integrated vehicle and crew scheduling approach. This is again done by solving an MIP with heuristic column generation. In the second-phase MIP, the tasks to be performed correspond to the vehicle routes computed in the first phase, and the decision variables correspond to crew schedules. An initial set of columns is created by restricted enumeration. A list is created containing one element for each possible start time of each task (vehicle route). The list is sorted by non-decreasing start time. Then, a depth-first recursive search algorithm starting with the first element in the list is used to create schedules. As many tasks as possible are added to a partial schedule. Afterwards, additional columns with negative reduced cost are determined by solving a resource-constrained shortest path problem (the authors do not give details on how this is done). When no more columns are found, an integer solution is determined by branch-and-bound.

The link between the crew and the vehicle schedules, that is, movement s., is ingeniously established in the following way: For each crew schedule, it is known which tasks (vehicle routes) the schedule fulfils, and, as stated above, for each vehicle route computed in the first phase, it is known with which vehicle type the route is performed. A constraint in the master problem then matches the number of vehicles of each type to the active crew schedule variables. The vehicle schedules are not given explicitly by a solution to the second-phase problem, rather, they are retrieved from such a solution by a simple procedure.
The multiple-depot aspect of the problem is addressed by introducing crew schedule variables for different combinations of start, intermediate, and end depots. For example, for a vehicle route $i \to j \to j \to i$ from the first phase with two requests $i \to j$ and $j \to i$, where i and j are depot locations, four variables can be introduced: First, a variable for the possibility that the complete schedule is performed by a vehicle whose home depot is i. Second, similarly, a variable for the possibility that the schedule is performed by a vehicle stationed at j. Third, a variable for the possibility that the first request is transported by a vehicle stationed at i and the second request is transported by a vehicle stationed at j. Fourth, a variable for the possibility that the first request is transported by a vehicle stationed at j and the second request is transported by a vehicle stationed at i. In this way, the number of possible schedule variables grows exponentially with the number of requests in a schedule. This number, however, is rather small for the problem instances considered by the authors.
Overall, the problem, the models, and the solution approach presented in the paper are highly complex and sophisticated. There are several additional aspects to the problem and its solution that could not be described here.

Drexl [52] presents formulations for the VRPTT based on turn, arc, and path variables. Assume for simplicity that there are only task vehicles (lorries and drawbar trailers), and that load transfers are possible at TLs from any lorry to any trailer.
Turn variables have their origin in arc routing problems with turn penalties, where performing a turn from one road segment into another at a road junction may be undesired or forbidden. [52] contains an extensive reference list on such problems. In a simple digraph, a *turn* is uniquely described by an ordered sequence of three vertices. If a vehicle traverses an arc (i, j) immediately after an arc (h, i), it performs a turn in vertex i, and this turn is denoted (h, i, j). A turn variable, hence, is a binary variable x^k_{hij} which equals one iff vehicle k performs the turn (h, i, j).
The turn variable formulation assumes that there is one vertex for each combination of TL and trailer and uses x^k_{hij} variables, binary $x^{kk'}_{ij}$ variables which equal one iff lorry k pulls trailer k' over arc (i, j), and two types of continuous time variable, t^k_{hi} indicating the time when k starts its operation at vertex i after having reached i via arc (h, i), and $t^{d,k}_{ij}$ indicating the time when k departs from i via (i, j). Movement s. is ensured by the $x^{kk'}_{ij}$ variables, which are linked to the turn variables by

$$\sum_{\{(h,i,j):(h,i),(i,j)\in A\}} x^{k'}_{hij} = \sum_{k\in F_L} x^{kk'}_{ij} \quad \forall\ k' \in F_T, (i, j) \in A \tag{13}$$

and

$$\sum_{\{(h,i,j):(h,i),(i,j)\in A\}} x^{k}_{hij} \geqq \sum_{k'\in F_T} x^{kk'}_{ij} \quad \forall\ k \in F_L, (i, j) \in A. \tag{14}$$

Moreover, it is required that $x^{kk'}_{ij} = 1$ imply that the arrival and departure times of k and k' at j be equal, and it is forbidden that at transshipment vertices i representing a location-trailer combination (l, k'), a trailer $k'' \neq k'$ be decoupled. Decoupling is also not allowed at customer vertices. This need not be explicitly required: A trailer must return to the depot. If a trailer were decoupled at a trailer customer, it could not be coupled again, because each customer is visited exactly once by exactly one lorry.

To ensure exact simultaneous operation s. at transshipment vertices, in addition to the trivial requirement that $x^k_{hij} = 1$ imply $t^k_{hi} \leqq t^{d,k}_{ij}$, constraint types are used stating that (i) a lorry k cannot perform a load transfer at a transshipment vertex i before the trailer k' belonging to i has arrived, and k' cannot receive any load from k before k has arrived and (ii) any load transfer at i must be finished by the time k' leaves i. Moreover, at any time, at most one lorry can transfer load to a trailer. If a lorry k_2 arrives at i while a lorry k_1 is performing a load transfer to k', k_2 must wait until k_1 is finished. To ensure this, it is required for any pair of lorries (k_1, k_2) that the beginning of a load transfer from k_1 to k' plus the load transfer time not overlap with the beginning of k_2's load transfer. This is modelled by a set of disjunctive constraints, using several additional types of variable.
The arc and path variable formulations are based on a network with at least two vertices for each combination of TL l and trailer k' representing the operations decoupling, transshipment, and coupling. To be precise, for each l and k', there are one decoupling vertex, zero or more transshipment vertices, and one coupling vertex. A compatible lorry can pull k' to the decoupling vertex, decouple there, and may also perform a load transfer. At the transshipment vertices, any lorry may transfer load to k'. At the coupling vertex, a compatible lorry may perform a load transfer, must couple k' and pull it away. k' moves on its own in time from the decoupling vertex to the first transshipment vertex, from there to the second one etc., and from the last transshipment vertex to the coupling vertex.
The arc variable formulation uses binary x^k_{ij} routing variables and continuous t^k_i time variables. Movement s. is then ensured by

$$x^{k'}_{ij} \leqq \sum_{k \in F_L} x^k_{ij} \quad \forall\ k' \in F_T, (i,j) \in A, \tag{15}$$

$$x^k_{hi} = 1 \Rightarrow t^{k'}_i \leqq t^k_i \quad \forall\ k' \in F_T, k \in F_L, (h,i) \in A, \tag{16}$$

and

$$x^{k'}_{hi} = 1 \Rightarrow t^k_i \leqq t^{k'}_i \quad \forall\ k' \in F_T, k \in F_L, (h,i) \in A. \tag{17}$$

Moreover, the above two constraint types ensure exact simultaneous operation s. at trailer customer or transshipment vertices i.
This approach has the drawback that, to linearize the above constraints, Big-M coefficients have to be introduced, weakening the LP-relaxation. (The turn variable formulation suffers from the same problem.)
The path variable formulation is based on the arc variable formulation and keeps the t^k_i variables and the above constraints in the master problem, which leads to similar difficulties with the solution of the pricing problems as in [76].
Computational experiments are performed with a branch-and-cut algorithm for the arc variable formulation. Several classes of valid inequalities and specialized branching strategies are developed. Nevertheless, only very small instances with up to 6 customers, 6 TLs, 2 lorries and 2 trailers can be consistently solved.

Berning [14] develops an MIP model for the simultaneous vehicle and crew routing and scheduling problem with pickups and deliveries in the context of advanced truckload transport. Drivers can change vehicles and vice versa at a discrete set of relay stations. A maximal daily working time, a maximal route duration, and a minimal duration for a rest between two routes are considered for the drivers. The rest must be taken at a relay station. Transshipment of load is not allowed.
The model is based on a network with one vertex for each pickup and delivery location of each request, one start and one end depot vertex for each driver and each vehicle, and n_l vertices for each relay station l, where n_l is an upper bound on the number of times l will be visited by any single driver or vehicle. The arc set is partitioned into spacial arcs between vertices representing different locations and virtual arcs between vertices corresponding to the same relay station.

Spacial arcs can only be traversed by a driver-vehicle combination, virtual arcs can be traversed by a driver without a vehicle and a vehicle without a driver. The virtual arcs allow a change of driver or vehicle at relay stations under the condition that each vertex of the network is visited at most once by any driver or vehicle. The network is similar to that for the arc variable formulation of the VRPTT described in [52].
Based on this network, the model essentially uses the following types of variable: Binary flow variables $x_{ij}^{kk'}$, which equal one iff driver k' drives vehicle k along arc (i, j), and continuous time variables t_i^k, $t_i^{k'}$ indicating the point in time when vehicle k, respectively, driver k' arrive at vertex i. The flow variables along virtual arcs also use four indices. The 'driver' of a vehicle moving along such an arc is a virtual one and vice versa.
Movement s. is achieved by the $x_{ij}^{kk'}$ variables and the following types of constraint:
A change of driver or vehicle is not allowed at pickup and delivery vertices i:

$$\sum_{(h,i)\in A} x_{hi}^{kk'} - \sum_{(i,j)\in A} x_{ij}^{kk'} = 0 \quad \forall\ k \in F, k' \in F_D \tag{18}$$

Arrival times of driver and vehicle at the head vertex of an arc traversed together must be equal:

$$x_{ij}^{kk'}(t_j^k - t_j^{k'}) = 0 \quad \forall\ k \in F, k' \in F_D, (i, j) \in A \tag{19}$$

Flow conservation for vehicles at relay station vertices i must be ensured:

$$\sum_{k'\in F_D} \sum_{(h,i)\in A} x_{hi}^{kk'} - \sum_{k'\in F_D} \sum_{(i,j)\in A} x_{ij}^{kk'} = 0 \quad \forall\ k \in F \tag{20}$$

There are analogous constraints for the drivers.
The model is implemented and validated by solving small random test instances with a standard MIP solver.

Bürckert et al. [24] describe a decision support system for a dynamic PDPTW in the context of long-distance transport. The optimization component of the system is based on the concept of holonic multi-agent systems.
The authors distinguish no less than eight types of object: driver, lorry with loading capacity, lorry without loading capacity, tractor, trailer, semi-trailer, chassis, and swap-body. Each concrete object is modelled as an agent. Moreover, holonic agents, that is, agentifications of representations for conglomerates of objects that together form a vehicle able to perform requests, are introduced. Holonic agents, or holons, differ from other groups of cooperating agents in that their members essentially act as a unit for a certain period of time. For each holon, there is a meta-agent that coordinates the formation of the holon and represents it to the outside world.
There are three modes of operation of the multi-agent system: (i) The insertion mode, where solutions are generated and extended by sequential insertion of requests into existing plans; (ii) the optimization mode, where existing plans are improved by exchanging orders between vehicles; and (iii) the interactive mode, where a human planner may change existing plans by hand.
The insertion mode works as follows. If a new request is to be performed, a meta-agent to construct a new holon as well as the meta-agents of all existing holons that are currently executing a route are informed. The former meta-agent starts a process called extended contract net protocol (ECNP). This means that the information about the new request is analyzed with respect to the elementary objects required to perform the request. Then, this information is handed on to the set of yet unused motorized objects/agents. Each motorized agent, in turn, checks whether it is capable of performing the request and which requirements regarding driver time and qualification and chassis/trailer/semi-trailer type and capacity it needs to do so. This information is handed on to the driver and chassis/trailer/semi-trailer agents, and these respond whether or not they are suitable for performing the request. The motorized agents evaluate this

information and inform the meta-agent, which will eventually form a holon that can fulfil the request. Each already existing holon checks whether it is able to perform the new request. If so, it computes the costs for performing the request. Otherwise, it also starts an ECNP process to acquire the missing capabilities/capacities.
The optimization mode the authors use is essentially a local search procedure on existing routes performed by fixed composite objects, embedded in a simulated annealing metaheuristic to escape local minima. Several iterations are performed, and in each iteration, each meta-agent removes at most one request from its route. These requests are then inserted into other routes.
The overall workings of the algorithm are described in a very high-level, general, abstract way. The enormously complicated details of operation s. in space and time between eight types of object are not even mentioned. It is imaginable that the selection of different types of object out of a candidate set and their aggregation to a meta-object that is able to perform one request is a tractable thing to do. And of course it is standard to check the feasibility of an insertion of a new request into the route of an existing composite object. However, it is surely a daunting enterprise to consider the multitude of different options of modifying existing composite objects in a sensible and comprehensive way. Most probably, the reconfiguration options must be strictly limited.
A remarkable feature of the paper is that the employed procedure is not based on a network, neither explicitly nor implicitly. The authors do not mention any details on this issue, and neither do they report any computational results, so that it is unclear what size instances can be solved with their approach.

Cheung et al. [30] describe an application of a pickup-and-delivery problem where containers must be transported from a seaport to inland destinations. There are three types of non-autonomous object which are required to transport a container: a driver, a tractor, and a semi-trailer. The authors develop an attribute-decision model and solve it heuristically by a labelling algorithm. As in [24], the developed solution procedure is not based on a network.
The basic idea is to represent an object by a set of attributes associated with a set of possible decisions. An object can either be an elementary (driver, tractor, semi-trailer, container) or a composite one (driver-tractor, tractor-semi-trailer, driver-tractor-semi-trailer etc.). The authors distinguish between active (autonomous) and passive (non-autonomous) objects. Only drivers are considered active objects. The attributes are either generic or object-type specific. Generic attributes are location, point in time at which the object is at the location, and overall time window of availability. Specific attributes are, for example, the remaining working time for a driver or the set of compatible semi-trailers for a tractor. For each object with a certain value of its attributes, there is a set of decisions that can be executed on the object. Basically, there are three types of decision: Couple, uncouple, and modify. Couple decisions represent the act of combining an object with another, for example, assigning a driver to a tractor. Uncouple decisions, of course, represent the opposite. Modify decisions change the attributes of an object without interaction with other objects. For example, the movement of a composite object driver-tractor-semi-trailer from one location to another changes the attributes location and time of the composite object. The only possible decision type for a driver is to couple a tractor. The possible decision types for a driver-tractor composite object are to couple a semi-trailer, to uncouple the tractor, and to move back to the driver home depot. For a driver-tractor-semi-trailer composite object, the possible decision types are to load a container, to move empty to another location, and to uncouple the semi-trailer. Finally, for a driver-tractor-semi-trailer-container composite object, the possible decision types are to unload the container, to load one more container (the loading capacity of a semi-trailer is either two small containers or one large container), to uncouple the semi-trailer together with the container(s), and to move to another location.
The major steps of the labelling algorithm are as follows. The attribute sets for all elementary objects are created. The drivers are sorted in non-decreasing order of earliest available time and are added to a candidate list. The first object in the list is selected, the set of possible decisions

for this object is determined, and the value of each decision is computed. This value is comprised of three components: a direct cost, the estimated cost for the use of any passive object(s) due to a decision, and the estimated future value of the object and its attributes resulting from the decision. The estimated future value is computed by a limited tree search approach that evaluates the effects of the decision. The decision with the highest value is executed. This means that the object resulting from the decision is added to the top of the candidate list. After that, the estimated costs for all passive resources involved in the decision are updated using the opportunity cost principle, that is, measuring the cost difference between the decision taken and the second best decision. After that, all previously made decisions are updated, considering the effects of the last decision made. This is repeated until the candidate list is empty or an iteration limit is reached. Depending on the instance, it is not guaranteed that all requests are performed. This algorithm is capable of solving instances of up to 40 drivers, 40 tractors, 40 semi-trailers and 240 requests.
Operation s. is achieved by the combination of (i) generating the set of possible decisions upon having selected an object with its current attributes from the candidate list and (ii) updating all previous decisions after a new decision has been taken. Movement s. is ensured elegantly by the fact that only objects comprised of at least a driver and a tractor can move in space.
This approach is very different from standard local or large neighbourhood search heuristics. It is surely a considerable challenge to implement this algorithm. The details of determining the set of possible decisions, of estimating the cost of using passive objects in a decision and of the future values of objects as well as of updating previous decisions may become extremely complex. Moreover, it is unclear how well the algorithm will work on instances with a different structure. On the other hand, the principal power of the concept and the algorithm make their application to other problems a very interesting and challenging research topic.

Concluding remarks on movement synchronization en route

The survey shows that, as with movement s. at the depot, the number of publications dealing with movement s. en route is limited, but the applications are again diverse, and so are the described solution approaches.
It is interesting to note that Imai et al. [75], who study an application similar to the one dealt with in [30], state (p. 88): 'A trailer-truck consists of a tractor and a trailer, and normally they can be uncoupled ... However, this is not likely the case especially for intermodal container transportation within Japan, mainly because of the complexity of the tractor assignment to trailer'. Scheuerer (2006, personal communication) states that his work on VRPs with trailers for raw milk collection ([112]) was also limited to a fixed lorry-trailer assignment due to the perceived difficulties of developing a heuristic for problems with free lorry-trailer assignment, that is, for the VRPTT. This supports the conjecture that the limited number of papers in this field is due to the difficulty of movement s. en route, not to the lack of practical applications or importance. Nevertheless, Bürckert et al. [24] and Cheung et al. [30] present approaches which may be modifiable to devise a heuristic for solving real-world VRPTTs.

6 Summary and Conclusion

This paper has surveyed vehicle routing problems with multiple synchronization constraints. A VRPMS has been defined as a VRP where more than one vehicle may or must be used to fulfil a task. The VRP with trailers and transshipments has been presented as an archetypal example of a VRPMS, and a classification of different types of synchronization requirements that appear in real-world vehicle routing problems has been developed. The decisive modelling and solution issues with VRPMSs have been pointed out: The interdependence problem encountered in VRPMSs, the fact that a change in one route may have effects on other routes, considerably complicates the use of standard solution techniques for VRPs, such as column generation and

local search. Most importantly, literature on synchronization has been surveyed, focussing on applications and on the techniques for dealing with the synchronization requirements.
With respect to *applications* or rather problem classes, the literature review has shown that there is an increasing number of papers on VRPMSs with transshipments and temporal s. of visits. Papers on synchronizing autonomous and non-autonomous objects are still rare. The *most important concrete problem classes* are (i) N-echelon VRPs/LRPs, (ii) PDPTW with transshipments, and (iii) simultaneous vehicle and crew routing and scheduling problems.
With respect to *modelling and design decisions* in papers presenting MIP approaches, it is very interesting to observe the different options for creating an MIP model, even though none of these options constitutes a silver bullet: The spectrum where the information, data, and relationships of a concrete problem, in particular, the synchronization requirements, are represented ranges from 'model the underlying logic completely by means of decision variables and constraints' to 'create a highly involved network that by itself ensures synchronization'. At the former end of the spectrum lie the works [83], [113], [126], [18]. Moving toward the latter end, the papers [107], [123], [38], [95], [68], and [88] can be located. [56] and [57] are close to the latter end, which is defined by [3] and [4].
With respect to *solution methods*, there are some *recurring algorithmic techniques* and principles which are used in several papers. These are indicated in the following tables.

Technique	References
Using one vehicle-independent time variable for the beginning of execution of a task or operation requiring more than one vehicle	[86], [87], [51], [38]
Discretizing time	[76], [12], [51], [68]
Branching on time windows	[76], [12], [51]
Explicitly determining request paths or flows and linking them to vehicle paths/flows	[41], [68], [95]

Table 6: Recurrent exact techniques

Technique	References
Decomposition by stage for 2-echelon VRPs/LRPs	[111], [40], [98], [15], [41]
Explicitly determining request leg sequences	[55], [100], [18], [2], [93]
Allowing intermediate infeasible solutions and penalizing them in the objective function	[42], [123], [100], [106]
Estimating/incompletely evaluating a neighbourhood move	[42], [123]
Indirect search	[86], [87], [55]
Decomposition by type of mobile object for movement s. of drivers and vehicles	[125], [126], [73]

Table 7: Recurrent heuristic techniques

All in all, this survey has demonstrated that multiple synchronization constraints are a challenging extension of VRPs, and that they are of practical relevance in many different application areas. Consequently, VRPMSs require and deserve to be studied further. The following final section of this survey outlines promising fields and directions for doing so.

7 Outlook

There are many fields and problems requiring some type or other of synchronization. Due to the limited scope of this survey, these fields and problems were not examined in detail here, but their further study may yield valuable insights for solving VRPMSs, too. Pertinent references, mostly survey papers, are given in the following table. A brief discussion follows.

Application	References
Strategic and tactical network design	Crainic/Kim [39], Wieberneit [124], Andersen et al. [6]
Intermodal transport	Macharis/Bontekoning [92], Caris et al. [27]
Integrated vehicle and crew scheduling	Klabjan [84], Caprara et al. [26]

(continued on next page)

(continued from previous page)

Application	References
Transit scheduling	Desaulniers/Hickman [49]
School bus routing	Park/Kim [102]
Maritime transport	Christiansen et al. [32], Hennig [70]
Periodic VRPs	Cordeau et al. [36], Angelelli/Speranza [8]
Inventory routing problems	Campbell et al. [25], Moin/Salhi [94], Andersson et al. [7], Oppen et al. [101]
Stochastic VRPs	Gendreau et al. [60], Rivers [108], Christiansen/Lysgaard [31], Liu/Lai [89]
Swapping problem	Anily/Hassin [9], Bordenave et al. [19]
VRPs with uncapacitated intermediate depots or refill stations	Ghiani et al. [62], Tarantilis et al. [119]
Single-echelon location-routing problems	Nagy/Salhi [97]
Truck-and-trailer routing problem (TTRP)	Semet/Taillard [115], Gerdessen [61], Chao [28], Scheuerer [112]

Table 8: Fields related to VRPMSs and pertinent literature

An important class of problems is *strategic and tactical network design*. These problems also have synchronization aspects, and, regularly, VRPs have to be solved as sub-problems. However the vehicle routing component, due to the strategic/tactical nature of network design, is often only addressed by coarse approximations.

Intermodal transport, by its very definition, requires the transshipment of load. However, the field is very wide and could therefore not be covered in detail. Some particularly relevant papers were discussed above.

With the exception of airline fleet assignment with schedule synchronization constraints, *integrated vehicle and crew scheduling* (as opposed to simultaneous vehicle and crew *routing and* scheduling) problems (in airline, railway, or public transport applications) were also not considered. Although these problems do require synchronization, the presence of given schedules for itineraries and/or vehicles and/or crews changes the nature of the problem and allows for solution approaches not directly applicable to vehicle *routing* problems. Moreover, there is a huge body of literature on these problems. Also in the related area of public *transit scheduling* (again, *scheduling* as opposed to *routing*), the existing body of literature is huge, and there are hundreds of papers considering problems with transfers (passengers changing trains etc.). In the context of *school bus routing*, Park/Kim [102] state (p. 318): 'The issue of student transshipment should ... be considered in future research.'

Vehicle routing problems also arise in *maritime transport*. A prominent example of exceeding economic importance is oil tanker routing and scheduling.

In *periodic VRPs*, synchronization between different vehicles is necessary in that the several visits required to serve a customer during the planning horizon, which must take place in different periods, may be performed by different vehicles. However, this interdependence of vehicles is essentially removed by the fact that periodic VRPs operate on visiting patterns for a customer instead of single visits. Once a decision on a visiting pattern is made, the visits this pattern consists of are independent. Therefore, periodic VRPs were also excluded here.

A very special type of VRP are *inventory routing problems (IRPs)*. These differ significantly from other VRPs in several respects. In IRPs, there are no customer demands. Instead, each customer has a given consumption rate of a good, a given initial stock and a given storage capacity. The depot has to perform zero or more deliveries to each customer during a multi-period planning horizon to ensure that no customer runs out of stock. The objective is to plan routes of minimal cost for the deliveries. IRPs require load s. at customers, induce temporal interdependencies between different routes, and fulfil the definition of a VRPMS as given in the Introduction. However, due to their special nature, a more detailed treatment of IRPs is beyond the scope of this paper. The reader is referred to the literature in the above table.

A special type of task s. appears in *stochastic VRPs*, that is, problems where the customer supplies (or demands) are not deterministic, but are given by a probability distribution or a

fuzzy number, and the concrete amount of supply/demand for a customer becomes known only when a vehicle reaches the customer. Then, the vehicle capacity may be insufficient to serve the customer. This is called a failure. To recover, the route planned a priori has to be changed. This is usually done in a non-cooperative way by returning to the depot and then returning to the customer where the failure occurred and finishing the route as planned. It can also be done in a cooperative way, where the customer-vehicle assignment is re-planned after a failure. This, however, does not cause multiple synchronization constraints and, hence, was not considered here.

Further VRPs which do not require multiple synchronization in the sense used here are the *swapping problem*, *VRPs with uncapacitated intermediate depots or refill stations*, and *single-echelon location-routing problems*.

Moreover, the *truck-and-trailer routing problem* (TTRP) is also not a VRPMS. The TTRP is a special case of the VRPTT where there is a fixed lorry-trailer assignment. This means that each trailer can be pulled by only one lorry, and only this lorry can transfer load into the trailer. Consequently, there are no support vehicles, and there is no interdependence problem.

With respect to *applications* of VRPMSs, it is to be expected that *integrated vehicle and crew routing and scheduling in road transport*, *city logistics*, *forest management*, and *agricultural field operations* will be more intensively studied and hence yield more publications in the near future. In particular, the last topic seems to offer a wide range of challenging applications, as the recent survey by Bochtis/Sørensen [16], [17] shows. These authors present several problems in agricultural field logistics which can be modelled as VRPs. Some of these problems contain multiple synchronization constraints. Most importantly, many of these problems require the use of task and support vehicles, and, consequently, of task, operation, and load s. Solution approaches or results of practical projects are not reported yet.

With respect to *algorithmic approaches*, two fields not directly connected to transport logistics, contrary to the areas described before, which nevertheless show great relevance for VRPMSs, are *scheduling* and *robotics*.

Some problems, for example the manpower assignment problems discussed in Section 5.3 bear some similarity to *machine or project scheduling* problems. In particular, resource-constrained project scheduling problems are related to VRPMSs with resource s. It was beyond the scope of this paper to survey the considerable body of literature on scheduling problems. A thorough introduction and an up-to-date overview are given by Brucker/Knust [22] and Brucker [21]. It is to be expected that ideas and principles used for the solution of scheduling problems can also be useful for solving VRPMSs and vice versa.

The same holds true for applications in *robotics and control theory*. The free textbook Bullo et al. [23] contains a very extensive reference list for further reading in this field. It seems that recently, similar to the situation in the vehicle routing and operational research literature, there is an increasing number of papers addressing problems of coordinating robots for performing interdependent tasks in space and time. Two particularly interesting journal papers are Shima et al. [117] and Jones et al. [82]. The latter paper also contains a substantial list of pertinent references.

A closer look at interfaces and commonalities of these three fields, VRPMSs, resource-constrained scheduling, and robot coordination, constitutes an interesting and promising research perspective.

Acknowledgement This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/5-1.

Appendix

Glossary of terms

Task, request	A mandatory duty, something which must be done and requires zero or more units of some capacity (collecting supply or delivering demand at one location, picking up load at one location and delivering this load to another location, visiting a location to render a service etc.).
Operation	Something that may or must be performed by a vehicle at a location or vertex (a transshipment, coupling or uncoupling a trailer etc.).
Autonomous vehicle	A vehicle able to move in time and space on its own.
Non-autonomous vehicle	A vehicle able to move in time on its own, but which must be accompanied by an autonomous vehicle or must join with one or more other non-autonomous vehicle(s) to be able to move in space.
Task vehicle	Task vehicles are allowed to visit customers.
Collection vehicle	Task vehicles in problems where supply has to be collected at the customers.
Delivery vehicle	Task vehicles in problems where demand has to be delivered to customers.
Support vehicle	Support vehicles are not allowed to visit customers and only act as mobile depots for the task vehicles. The term was independently coined by [108] and [52].
Active vehicle	The vehicle transferring load in a transshipment.
Passive vehicle	The vehicle receiving load in a transshipment.
Lead vehicle	The vehicle determining the location and the time for a transshipment.
Dynamic time window	Time window for execution of an operation; determined in the course of a solution procedure; depends on the execution of another operation.
Interdependence problem	Fact that in VRPMSs, vehicle routes may depend on one another, so that changes in one route affect other routes.
Path, itinerary	Sequence of locations or vertices visited by a vehicle or a request on its way from its origin (depot) to its destination (depot).
Leg	Ordered pair of locations or vertices a request or vehicle must visit in the specified order. Other locations or vertices may be visited in between. In problems with transshipments, a request may only be transshipped between legs; on one leg, a request is always transported by the same vehicle.
Request leg sequence	Sequence of legs where the first location or vertex of the first leg is the origin of the request, the first location or vertex of all other legs is the second location or vertex of the previous leg, and the second location or vertex of the last leg is the destination of the request.

Summary of abbreviations and notation

VRP(TW)	vehicle routing problem (with time windows)
VRPMS	VRP with multiple synchronization constraints
VRPTT	VRP with trailers and transshipments
PDP(TW)	pickup-and-delivery problem (with time windows)
LRP	location-routing problem
(C)ARP	(capacitated) arc routing problem
DARP	dial-a-ride problem
(E)SPPRC	(elementary) shortest path problem with resource constraints
LP	linear program(ming)
MIP	mixed integer program(ming)

(continued on next page)

(continued from previous page)

BCP	branch-cut-price
s.	synchronization
TL	transshipment location
iff	if and only if
$A \cup B$	disjoint union of sets A and B
L	set of real-world locations of depots, customers, TLs etc.
$D = (V, A)$	digraph or network D with vertex set V and arc set A
0	depot location or vertex
o, d	start or origin and end or destination depot location or vertex
h, i, j	locations or vertices
(i, j)	arc from i to j; all considered graphs are simple without loss of generality
t_{ij}	traversal time of arc (i, j)
F	overall vehicle fleet or set of autonomous objects, indexed by k
F'	subset of overall vehicle fleet or set of non-autonomous objects, indexed by k'
F_L	set of lorries and/or tractors (autonomous vehicles), indexed by k
F_T, F_D	sets of trailers and drivers respectively, indexed by k' (non-autonomous vehicles)
Q, Q^k	capacity of a vehicle, capacity of vehicle k
R	set of tasks or requests
$r^+ \to r^-$	task or request r with pickup and delivery location or vertex r^+ and r^-; request leg from r^+ to r^-
T	length of planning horizon
P	set of paths or routes in a column generation approach
x_{ij}	binary routing variable equalling one iff vertex i is visited directly before vertex j
x_{ij}^k	binary routing variable equalling one iff vehicle k visits vertex i directly before vertex j
λ_p^k	binary path variable equalling one iff vehicle k performs path p
t_i^k	time variable indicating the beginning of execution of task i or the visiting time of vertex i by vehicle k
t_i	time variable indicating the beginning of execution of task i or the visiting time of vertex i

References

[1] Ahuja R, Ergun O, Orlin J, Punnen A (2002):
A Survey of Very Large-Scale Neighborhood Search Techniques
Discrete Applied Mathematics 123: 75–102

[2] Aldaihani M, Dessouky M (2003):
Hybrid Scheduling Methods for Paratransit Operations
Computers & Industrial Engineering 45: 75–96

[3] Amaya A, Langevin A, Trépanier M (2007):
The Capacitated Arc Routing Problem with Refill Points
Operations Research Letters 35: 45–53

[4] Amaya C, Langevin A, Trépanier M (2010):
A Heuristic Method for the Capacitated Arc Routing Problem with Refill Points and Multiple Loads
Journal of the Operational Research Society 61: 1095–1103

[5] Ambrosino D, Scutellà M (2005):
Distribution Network Design: New Problems and Related Models
European Journal of Operational Research 165: 610–624

[6] Andersen J, Crainic T, Christiansen M (2009):
Service Network Design with Management and Coordination of Multiple Fleets
European Journal of Operational Research 193: 377–389

[7] Andersson H, Hoff A, Christiansen M, Hasle G, Løkketangen A (2010):
Industrial Aspects and Literature Survey: Combined Inventory Management and Routing
Computers & Operations Research 37: 1515–1536

[8] Angelelli E, Speranza M (2002):
The Periodic Vehicle Routing Problem with Intermediate Facilities
European Journal of Operational Research 137: 233–247

[9] Anily S, Hassin R (1992):
The Swapping Problem
Networks 22: 419–433

[10] Archetti C, Speranza M (2008):
The Split Delivery Vehicle Routing Problem: A Survey
in:
Golden B, Raghavan S, Wasil E (eds):
The Vehicle Routing Problem: Latest Advances and New Challenges
Springer, New York
103–122

[11] Baker H, Franz L, Sweigart J (1993):
Coordinated Transportation Systems: An Alternative Approach to Traditional Independent Systems
European Journal of Operational Research 66: 341–352

[12] Bélanger N, Desaulniers G, Soumis F, Desrosiers J (2006):
Periodic Airline Fleet Assignment with Time Windows, Spacing Constraints and Time Dependent Revenues
European Journal of Operational Research 175: 1754–1766

[13] Benders J (1962):
Partitioning Procedures for Solving Mixed-Variables Programming Problems
Numerische Mathematik 4: 238–252

[14] Berning B (2009):
Die optimale Verplanung von Transporten bei interdependenten Ressourcen am Beispiel des Advanced-Truckload-Konzeptes
Master's Thesis, School of sciences, University of Erlangen-Nuremberg

[15] Boccia M, Crainic T, Sforza A, Sterle C (2010):
A Metaheuristic for a Two Echelon Location-Routing Problem
in:
Festa P (ed):
Experimental Algorithms
Lecture Notes in Computer Science 6049
Springer, Berlin
288–301

[16] Bochtis D, Sørensen C (2009):
The Vehicle Routing Problem in Field Logistics Part I
Biosystems Engineering 104: 447–457

[17] Bochtis D, Sørensen C (2010):
The Vehicle Routing Problem in Field Logistics: Part II
Biosystems Engineering 105: 180–188

[18] Bock S (2010):
Real-time Control of Freight Forwarder Transportation Networks by Integrating Multimodal Transport Chains
European Journal of Operational Research 200: 733–746

[19] Bordenave C, Gendreau M, Laporte G (2010):
Heuristics for the Mixed Swapping Problem
Computers & Operations Research 37: 108–114

[20] Bredström D, Rönnqvist M (2008):
Combined Vehicle Routing and Scheduling with Temporal Precedence and Synchronization Constraints
European Journal of Operational Research 191: 19–29

[21] Brucker P (2007):
Scheduling Algorithms
Springer, Berlin

[22] Brucker P, Knust S (2006):
Complex Scheduling
Springer, Berlin

[23] Bullo F, Cortés J, Martínez S (2009):
Distributed Control of Robotic Networks
Princeton University Press, Princeton

[24] Bürckert H, Fischer K, Vierke G (2000):
Holonic Transport Scheduling with TELETRUCK
Applied Artificial Intelligence 14: 697–725

[25] Campbell A, Clarke L, Kleywegt A, Savelsbergh M (1998):
The Inventory Routing Problem
in:
Crainic T, Laporte G (eds):
Fleet Management and Logistics
Kluwer, Boston
95–113

[26] Caprara A, Kroon L, Monaci M, Peeters M, Toth P (2007):
Passenger Railway Optimization
in:
Barnhart C, Laporte G (eds):
Transportation
Handbooks in Operations Research and Management Science 14
Elsevier, Amsterdam
129–187

[27] Caris A, Macharis C, Janssens G (2008):
Planning Problems in Intermodal Freight Transport: Accomplishments and Prospects
Transportation Planning and Technology 31: 277–302

[28] Chao I (2002):
A Tabu Search Method for the Truck and Trailer Routing Problem
Computers & Operations Research 29: 33–51

[29] Chen S, Golden B, Wasil E (2007):
The Split Delivery Vehicle Routing Problem: Applications, Algorithms, Test Problems, and Computational Results
Networks 49: 318–329

[30] Cheung R, Shi N, Powell W, Simao H (2008):
An Attribute-Decision Model for Cross-Border Drayage Problem
Transportation Research Part E 44: 217–234

[31] Christiansen C, Lysgaard J (2007):
A Branch-and-Price Algorithm for the Capacitated Vehicle Routing Problem with Stochastic Demands
Operations Research Letters 35: 773–781

[32] Christiansen M, Fagerholt K, Nygreen B, Ronen D (2007):
Maritime Transportation
in:
Barnhart C, Laporte G (eds):
Transportation
Handbooks in Operations Research and Management Science 14
Elsevier, Amsterdam
189–284

[33] Christofides N, Eilon S (1969):
An Algorithm for the Vehicle-Dispatching Problem
Operational Research Quarterly 20: 309–318

[34] Chung H, Norback J (1991):
A Clustering and Insertion Heuristic Applied to a Large Routeing Problem in Food Distribution
Journal of the Operational Research Society 42: 555–564

[35] Codato G, Fischetti M (2006):
Combinatorial Benders' Cuts for Mixed-Integer Linear Programming
Operations Research 54: 756–766

[36] Cordeau J, Gendreau M, Laporte G (1997):
A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems
Networks 30: 105–119

[37] Cordeau J, Laporte G, Mercier A (2001):
A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows
Journal of the Operational Research Society 52: 928–936

[38] Cortés C, Matamala M, Contardo C (2010):
The Pickup and Delivery Problem with Transfers: Formulation and a Branch-and-Cut Solution Method
European Journal of Operational Research 200: 711–724

[39] Crainic T, Kim K (2007):
Intermodal Transportation
in:
Barnhart C, Laporte G (eds):
Transportation
Handbooks in Operations Research and Management Science 14
Elsevier, Amsterdam
467–537

[40] Crainic T, Mancini S, Perboli G, Tadei R (2010):
Multi-Start Heuristics for the Two-Echelon Vehicle Routing Problem
Technical Report CIRRELT-2010-30, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport

[41] Crainic T, Ricciardi N, Storchi G (2009):
Models for Evaluating and Planning City Logistics Systems
Transportation Science 43: 432–454

[42] De Rosa B, Improta G, Ghiani G, Musmanno R (2002):
The Arc Routing and Scheduling Problem with Transshipment
Transportation Science 36: 302–313

[43] Del Pia A, Filippi C (2006):
A Variable Neighborhood Descent Algorithm for a Real Waste Collection Problem with Mobile Depots
International Transactions in Operational Research 13: 125–141

[44] Derigs U, Döhmer T (2008):
Indirect Search for the Vehicle Routing Problem with Pickup and Delivery and Time Windows
OR Spectrum 30: 149–165

[45] Derigs U, Li B, Vogel U (2010):
Local Search-Based Metaheuristics for the Split Delivery Vehicle Routing Problem
Journal of the Operational Research Society 61: 1356–1364

[46] Desaulniers G (2010):
Branch-and-Price-and-Cut for the Split-Delivery Vehicle Routing Problem with Time Windows
Operations Research 58: 179–192

[47] Desaulniers G, Desrosiers J, Ioachim I, Solomon M, Soumis F, Villeneuve D (1998):
A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems
in:
Crainic T, Laporte G (eds):
Fleet Management and Logistics
Kluwer, Boston
57–93

[48] Desaulniers G, Desrosiers J, Solomon M (eds) (2005):
Column Generation
Springer, New York

[49] Desaulniers G, Hickman M (2007):
Public Transit
in:
Barnhart C, Laporte G (eds):
Transportation
Handbooks in Operations Research and Management Science 14
Elsevier, Amsterdam
69–127

[50] Desaulniers G, Villeneuve D (2000):
The Shortest Path Problem with Time Windows and Linear Waiting Costs
Transportation Science 34: 312–319

[51] Dohn A, Kolind E, Clausen J (2009):
The Manpower Allocation Problem with Time Windows and Job-Teaming Constraints: A Branch-and-Price Approach
Computers & Operations Research 36: 1145–1157

[52] Drexl M (2007):
On Some Generalized Routing Problems
PhD Thesis, Faculty of Business and Economics, RWTH Aachen University

[53] Ebben M, van der Heijden M, van Harten A (2005):
Dynamic Transport Scheduling under Multiple Resource Constraints
European Journal of Operational Research 167: 320–335

[54] El Hachemi N, Gendreau M, Rousseau L (2010):
A Hybrid Constraint Programming Approach to the Log-Truck Scheduling Problem
Annals of Operations Research
DOI 10.1007/s10479-010-0698-x

[55] Feige D (2003):
Kapazitiertes Nonbipartites Matching als Optimierungskern eines Planungsverfahrens für Logistikdienstleister in der Transportwirtschaft
Technical Report Nr. 2/2003, Diskussionsbeiträge zu Wirtschaftsinformatik und Operations Research, Wirtschaftswissenschaftliche Fakultät, Martin-Luther-Universität Halle-Wittenberg, S. 9–14

[56] Fügenschuh A (2006):
The Vehicle Routing Problem with Coupled Time Windows
Central European Journal of Operations Research 14: 157–176

[57] Fügenschuh A (2009):
Solving a School Bus Scheduling Problem with Integer Programming
European Journal of Operational Research 193: 867–884

[58] Funke B, Grünert T, Irnich S (2005):
Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration
Journal of Heuristics 11: 267–306

[59] Gélinas S, Desrochers M, Desrosiers J, Solomon M (1995):
A New Branching Strategy for Time Constrained Routing Problems with Application to Backhauling
Annals of Operations Research 61: 91–109

[60] Gendreau M, Laporte G, Séguin R (1996):
Stochastic Vehicle Routing
European Journal of Operational Research 88: 3–12

[61] Gerdessen J (1996):
Vehicle Routing Problem with Trailers
European Journal of Operational Research 93: 135–147

[62] Ghiani G, Improta G, Laporte G (2001):
The Capacitated Arc Routing Problem with Intermediate Facilities
Networks 37: 134–143

[63] Glover F, Kochenberger G (eds) (2003):
Handbook of Metaheuristics
Kluwer, Boston

[64] Golden B, Raghavan S, Wasil E (eds) (2008):
The Vehicle Routing Problem: Latest Advances and New Challenges
Operations Research/Computer Science Interfaces Series 43
Springer, Berlin

[65] Gonzalez-Feliu J (2009):
The Multi-Echelon Location-Routing Problem: Concepts and Methods for Tactical and Operational Planning
Technical Report, Laboratoire d'Economie des Transports, Institut des Sciences de l'Homme

[66] Gonzalez Feliu J, Perboli G, Tadei R, Vigo D (2008):
The Two-Echelon Capacitated Vehicle Routing Problem
Technical Report, Control and Computer Engineering Department, Politecnico di Torino, Italy

[67] Gørtz I, Nagarajan V, Ravi R (2009):
Minimum Makespan Multi-Vehicle Dial-A-Ride
in:
Fiat A, Sanders P (eds):
Algorithms—ESA 2009
Lecture Notes in Computer Science 5757
Springer, Berlin
540–552

[68] Grünert T, Sebastian H (2000):
Planning Models for Long-Haul Operations of Postal and Express Shipment Companies
European Journal of Operational Research 122: 289–309

[69] Hempsch C, Irnich S (2008):
Vehicle Routing Problems with Inter-Tour Resource Constraints
in:
Golden B, Raghavan S, Wasil E (eds):
The Vehicle Routing Problem: Latest Advances and New Challenges
Springer, New York
421–444

[70] Hennig F (2010):
Optimization in Maritime Transportation: Crude Oil Tanker Routing and Scheduling
PhD Thesis, Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology

[71] Hertz A, Laporte G, Mittaz M (2000):
A Tabu Search Heuristic for the Capacitated Arc Routing Problem
Operations Research 48: 129–135

[72] Hertz A, Mittaz M (2001):
A Variable Neighborhood Descent Algorithm for the Undirected Capacitated Arc Routing Problem
Transportation Science 35: 425–434

[73] Hollis B, Forbes M, Douglas B (2006):
Vehicle Routing and Crew Scheduling for Metropolitan Mail Distribution at Australia Post
European Journal of Operational Research 173: 133–150

[74] Hooker J, Natraj N (1995):
Solving a General Routing and Scheduling Problem by Chain Decomposition and Tabu Search
Transportation Science 29: 30–44

[75] Imai A, Nishimura E, Current J (2007):
A Lagrangian Relaxation-Based Heuristic for the Vehicle Routing with Full Container Load
European Journal of Operational Research 176: 87–105

[76] Ioachim I, Desrosiers J, Soumis F, Bélanger N (1999):
Fleet Assignment and Routing with Schedule Synchronization Constraints
European Journal of Operational Research 119: 75–90

[77] Ioachim I, Gélinas S, Soumis F, Desrosiers J (1998):
A Dynamic Programming Algorithm for the Shortest Path Problem with Time Windows and Linear Node Costs
Networks 31: 193–204

[78] Irnich S (2008):
A Unified Modeling and Solution Framework for Vehicle Routing and Local Search-Based Metaheuristics
INFORMS Journal on Computing 20: 270–287

[79] Irnich S, Desaulniers G (2005):
Shortest Path Problems with Resource Constraints
in:
Desaulniers G, Desrosiers J, Solomon M (eds):
Column Generation
Springer, New York
33–65

[80] Jacobsen S, Madsen O (1980):
A Comparative Study of Heuristics for a Two-Level Routing-Location Problem
European Journal of Operational Research 5: 378–387

[81] Jepsen M, Petersen B, Spoorendonk S (2008):
A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with a Capacity Constraint
Technical Report 08/01, Department of Computer Science, University of Copenhagen

[82] Jones E, Dias M, Stentz A (2010):
Time-Extended Multi-Robot Coordination for Domains with Intra-Path Constraints
Autonomous Robots
DOI 10.1007/s10514-010-9202-3

[83] Kim B, Koo J, Park J (2010):
The Combined Manpower-Vehicle Routing Problem for Multi-Staged Services
Expert Systems with Applications 37: 8424–8431

[84] Klabjan D (2005):
Large-Scale Models in the Airline Industry
in:
Desaulniers G, Desrosiers J, Solomon M (eds):
Column Generation
Springer, New York
163–195

[85] Laurent B, Hao J (2007):
Simultaneous Vehicle and Driver Scheduling: A Case Study in a Limousine Rental Company
Computers & Industrial Engineering 53: 542–558

[86] Li Y, Lim A, Rodrigues B (2005):
Manpower Allocation with Time Windows and Job-Teaming Constraints
Naval Research Logistics 52: 302–311

[87] Lim A, Rodrigues B, Song L (2004):
Manpower Allocation with Time Windows
Journal of the Operational Research Society 55: 1178–1186

[88] Lin C (2008):
A Cooperative Strategy for a Vehicle Routing Problem with Pickup and Delivery Time Windows
Computers & Industrial Engineering 55: 766–782

[89] Liu C, Lai M (2009):
The Vehicle Routing Problem with Uncertain Demand at Nodes
Transportation Research Part E 45: 517–524

[90] Lu Q, Dessouky M (2006):
A New Insertion-Based Construction Heuristic for Solving the Pickup and Delivery Problem with Time Windows
European Journal of Operational Research 175: 672–687

[91] Lübbecke M, Desrosiers J (2005):
Selected Topics in Column Generation
Operations Research 53: 1007–1023

[92] Macharis C, Bontekoning Y (2004):
Opportunities for OR in Intermodal Freight Transport Research: A Review
European Journal of Operational Research 153: 400–416

[93] Mitrović-Minić S, Laporte G (2006):
The Pickup and Delivery Problem with Time Windows and Transshipment
INFOR 44: 217–227

[94] Moin N, Salhi S (2007):
Inventory Routing Problems: A Logistical Overview
Journal of the Operational Research Society 58: 1185–1194

[95] Mues C, Pickl S (2005):
Transshipment and Time Windows in Vehicle Routing
in:
Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks
IEEE, Piscataway
113–119

[96] Nagy G, Salhi S (1996):
Nested Heuristic Methods for the Location-Routeing Problem
Journal of the Operational Research Society 47: 1166–1174

[97] Nagy G, Salhi S (2007):
Location-Routing: Issues, Models and Methods
European Journal of Operational Research 177: 649–672

[98] Nguyen V, Prins C, Prodhon C (2010):
A Multi-Start Evolutionary Local Search for the Two-Echelon Location Routing Problem
in:
Blesa M, Blum C, Raidl G, Roli A, Samples M (eds):
Hybrid Metaheuristics
Lecture Notes in Computer Science 6373
Springer, Berlin
88–102

[99] Nowak M, Ergun O, White C (2008):
Pickup and Delivery with Split Loads
Transportation Science 42: 32–43

[100] Oertel P (2000):
Routing with Reloads
PhD Thesis, Faculty of Mathematics and Natural Sciences, University of Cologne

[101] Oppen J, Løkketangen A, Desrosiers J (2010):
Solving a Rich Vehicle Routing and Inventory Problem Using Column Generation
Computers & Operations Research 37: 1308–1317

[102] Park J, Kim B (2010):
The School Bus Routing Problem: A Review
European Journal of Operational Research 202: 311–319

[103] Perboli G, Tadei R, Vigo D (2008):
The Two-Echelon Capacitated Vehicle Routing Problem: Models and Math-Based Heuristics
Technical Report CIRRELT-2008-55, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport

[104] Perboli G, Tadei R, Masoero F (2009):
Valid Inequalities for the Two-Echelon Capacitated Vehicle Routing Problem
Technical Report CIRRELT-2009-39, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport

[105] Perboli G, Tadei R, Tadei R (2010):
New Families of Valid Inequalities for the Two-Echelon Vehicle Routing Problem
Electronic Notes in Discrete Mathematics 36: 639–646

[106] Prescott-Gagnon E, Desaulniers G, Rousseau L (2010):
Propane Delivery Vehicle Routing Problem
Technical Report, École Polytechnique de Montréal and GERAD

[107] Recker W (1995):
The Household Activity Pattern Problem: General Formulation and Solution
Transportation Research Part B 29: 61–77

[108] Rivers C (2002):
Coordination in Vehicle Routing
PhD Thesis, Massey University

[109] Ropke S (2005):
Heuristic and Exact Algorithms for Vehicle Routing Problems
PhD Thesis, Department of Computer Science, University of Copenhagen

[110] Russell R (1977):
An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Conditions
Operations Research 25: 517–524

[111] Russell R, Morrel R (1986):
Routing Special-Education School Buses
Interfaces 16: 56–64

[112] Scheuerer S (2006):
A Tabu Search Heuristic for the Truck and Trailer Routing Problem
Computers & Operations Research 33: 894–909

[113] Schmid V, Doerner K, Hartl R, Salazar-González J (2010):
Hybridization of Very Large Neighborhood Search for Ready-Mixed Concrete Delivery Problems
Computers & Operations Research 37: 559–574

[114] Schönberger J, Kopfer H, Wenning B, Rekersbrink H (2009):
Vehicle and Commodity Flow Synchronization
in:
Fleischmann B, Borgwardt K, Klein R, Tuma A (eds):
Operations Research Proceedings 2008
Springer, Berlin
307–312

[115] Semet F, Taillard E (1993):
Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search
Annals of Operations Research 41: 469–488

[116] Shang J, Cuff C (1996):
Multicriteria Pickup and Delivery Problem with Transfer Opportunity
Computers & Industrial Engineering 30: 631–645

[117] Shima T, Rasmussen S, Sparks A, Passino K (2006):
Multiple Task Assignments for Cooperating Uninhabited Aerial Vehicles Using Genetic Algorithms
Computers & Operations Research 33: 3252–3269

[118] Solomon M (1987):
Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints
Operations Research 35: 254–265

[119] Tarantilis C, Zachariadis E, Kiranoudis C (2008):
A Hybrid Guided Local Search for the Vehicle-Routing Problem with Intermediate Replenishment Facilities
INFORMS Journal on Computing 20: 154–168

[120] Thangiah S, Fergany A, Awan S (2007):
Real-Time Split-Delivery Pickup and Delivery Time Window Problems with Transfers
Central European Journal of Operations Research 15: 329–349

[121] Toth P, Vigo D (eds) (2002):
The Vehicle Routing Problem
SIAM Monographs on Discrete Mathematics and Applications, Philadelphia

[122] Tuzun D, Burke L (1999):
A Two-Phase Tabu Search Approach to the Location Routing Problem
European Journal of Operational Research 116: 87–99

[123] Wen M, Larsen J, Clausen J, Cordeau J, Laporte G (2008):
Vehicle Routing with Cross-Docking
Journal of the Operational Research Society 60: 1708–1718

[124] Wieberneit N (2008):
Service Network Design for Freight Transportation: A Review
OR Spectrum 30: 77–112

[125] Xiang Z, Chu C, Chen H (2006):
A Fast Heuristic for Solving a Large-Scale Static Dial-A-Ride Problem under Complex Constraints
European Journal of Operational Research 174: 1117–1139

[126] Zäpfel G, Bögl M (2008):
Multi-Period Vehicle Routing and Crew Scheduling with Outsourcing Options
International Journal of Production Economics 113: 980–996