Available online at www.sciencedirect.com

SCIENCE DIRECT®

Transportation Research Part B 40 (2006) 265–284

TRANSPORTATION RESEARCH PART B

www.elsevier.com/locate/trb

A shortest path approach to the multiple-vehicle routing problem with split pick-ups

Chi-Guhn Lee [a,*], Marina A. Epelman [b], Chelsea C. White III [c], Yavuz A. Bozer [b]

[a] *Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada*
[b] *Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, United States*
[c] *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, United States*

Received 8 October 2003; received in revised form 11 July 2004; accepted 22 November 2004

Abstract

We consider a multiple-vehicle routing problem with split pick-ups (mVRPSP). This problem involves multiple suppliers, a single depot, and a fleet of identical capacity trucks responsible for delivering supplies from the suppliers to the depot. Any supplier may be visited by more than one truck, thus allowing split pick-ups. The problem is to determine, for each truck, which suppliers to visit and the size of loads to pick up so as to minimize the total transportation cost for the fleet, which depends on the number of trucks used and their routes. We develop a fundamentally new model for the mVRPSP, a deterministic dynamic program (DP). Although the most natural DP formulation results in a DP with uncountably-infinite state and action spaces, an optimality-invariance condition we establish leads to an equivalent DP with finite state and action spaces. This DP formulation leads to a new exact algorithm for solving the mVRPSP, based on a shortest path search algorithm, which is conceptually simple and easy to implement.
Crown Copyright © 2005 Published by Elsevier Ltd. All rights reserved.

Keywords: Vehicle routing; Split deliveries/pick-ups; Dynamic programming; Shortest path algorithms

1. Introduction

Brought on by competition, the pressure to reduce cost is ever-present in the trucking industry. Often cost reduction results from improving operational processes, such as fleet routing. In this paper we consider a process-related problem of broad and significant interest to the transportation industry; the problem requires the decision-maker to

(1) determine the number of trucks needed to transport supplies (e.g., parts, modules, components, material) from a set of suppliers to a single depot (e.g., assembly plant), and

[*] Corresponding author.
E-mail addresses: cglee@mie.utoronto.ca (C.-G. Lee), mepelman@umich.edu (M.A. Epelman), cwhite@isye.gatech.edu (C.C. White III), yabozer@umich.edu (Y.A. Bozer).

0191-2615/$ - see front matter Crown Copyright © 2005 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.trb.2004.11.004

(2) for each truck, determine the suppliers it should visit and the amount of supplies it should pick up at each supplier, in order to minimize the total transportation cost.

We allow more than one truck to visit each supplier; i.e., we allow *split pick-ups*. We refer to this problem as the multiple vehicle routing problem with split pick-ups (mVRPSP). Dror and Trudeau (1989) have shown that the advantages of permitting split pick-ups can be significant, although allowing split pick-ups greatly adds to problem complexity, as we discuss in the following section. The problem has traditionally been modelled and solved as a Mixed Integer Program (MIP). Due to the problem complexity, however, it is necessary to use heuristic or approximate solution techniques when solving the majority of instances of the mVRPSP; we discuss the literature on such techniques below.

In this paper we present a new concept of optimality-invariance for this problem, and a new approach, formulating it as a deterministic dynamic program (DP). This approach was developed as a result of a collaboration with Ryder Integrated Logistics (RIL), a third-party logistics provider. As our discussions with RIL personnel indicated, when a logistics provider is responsible for transporting supplies from the same set of suppliers to the depot on a regular basis, truck dispatchers do not change the routing decision made for one such instance of the mVRPSP when the supply amounts vary slightly. In this paper, we describe the concept of *optimality-invariance*, i.e., we precisely characterize which variations in the supply amounts will not change the optimal collection of truck routes that should be used.

The second contribution of this paper is an alternative model for the mVRPSP, which we formulate as a DP. Although the most natural such formulation for the mVRPSP entails uncountably-infinite state and action spaces, we show, using the concept of optimality-invariance, that it is possible to modify the formulation to obtain a DP with finite state and action spaces. Based on the above formulation, we present a new, exact, algorithm for solving the mVRPSP. The algorithm is a particular implementation of a shortest path search algorithm in the directed network associated with the finite state DP above, and hence is conceptually simple and easy to implement.

We prove that the solution procedure we present is guaranteed to produce an optimal solution to the mVRPSP upon termination. In addition, we present some encouraging, albeit preliminary, computational examples. Although, in its greatest generality, the approach has a drawback typical of a DP algorithm, the "curse of dimensionality," we discuss why this method represents a promising new approach for a variety of practical instances of the mVRPSP.

Dror and Trudeau (1990) proposed a heuristic algorithm for the mVRPSP and identified certain properties of an optimal solution; this paper, as well as many of the papers cited below, rely on these properties. Dror et al. (1994) strengthened the MIP formulation of the mVRPSP by establishing several families of valid inequalities. They outlined a two-part branch-and-bound procedure, which makes use of these valid inequalities both to perform branching and to obtain tighter lower bounds on the value of the optimal solution to be used in the branch-and-bound algorithms. Although a full implementation of their algorithm would constitute a complete branch-and-bound procedure, and hence will eventually terminate with an optimal solution, the authors chose to implement only the first part of the algorithm (essentially, solving the relaxed problem at the root of the branch-and-bound tree), since their primary goal was to investigate the strength of the derived valid inequalities. They report gaps from 0% to 9% between the upper and lower bounds on the optimal objective value, where the gap of 0% was obtained in one problem instance with 10 suppliers. To our knowledge, the complete, and hence exact, algorithm has not been implemented. Hence, feasible solutions of the mVRPSP cannot be inferred from the obtained output.

Belenguer et al. (2000) studied the facial structure of the mVRPSP polyhedron, and also identified several classes of valid inequalities. Based on a heuristic procedure for the identification of violated constraints, they presented a cutting-plane algorithm which identifies a lower bound on the optimal objective value. They considered instances with up to 48 suppliers, and reported gaps between 0% and 12%. Once again, a feasible solution of the mVRPSP cannot in general be obtained from the output of this algorithm. To our knowledge, there has been no numerical evaluation of MIP-based solution procedures producing solutions for the mVRPSP that are provably optimal, which is the main feature of the approach we present in this paper.

Other heuristic algorithms for the mVRPSP were presented in Frizzell and Griffin (1995) and Sierksma and Tijssen (1998). Ho and Haugland (2004) present a tabu-search heuristic for the mVRPSP with time window constraints on visits to the suppliers.

It is interesting to note that the majority of computational experiments performed in the above papers considered instances of the mVRPSP in which each supplier had less than a truckload of supplies to be picked up. One exception is the paper by Sierksma and Tijssen (1998), where the authors argued that the approach in which dedicated routes are used to reduce the amount of supplies to less than a truckload in every location can lead to sub-optimal solutions when side constraints are imposed on the possible routes (in their example, by an upper bound on the length of a route). In this paper, the supply amounts may exceed one truckload.

In contrast with the traditional MIP formulations, we propose an entirely new way to approach the mVRPSP, namely, we model the mVRPSP as a dynamic program (DP). The state of this DP is the vector of supplies $a = (a_1, \ldots, a_M)$, where M is the number of suppliers, and the non-negative real number a_i represents the number of truckloads of supplies that needs to be transported from supplier i to the assembly plant. For example, if $a = (1.4, 0.5)$, then there are two suppliers; supplier 1 has 1.4 truckloads for pick-up, and supplier 2 has 0.5 truckloads for pick-up. The resulting state space is the non-negative orthant of $\mathbb{R}^M$.

Given a supply vector a, an action for this DP is a set of suppliers to be visited and the sizes of loads, i.e., number of truckloads of supplies to be picked up from these suppliers, by a single (capacitated) truck. We represent such an action by a non-negative vector $w = (w_1, \ldots, w_M)$ such that $w \leqslant a$ and $\sum_{i=1}^{M} w_i \leqslant 1$. The set of all such actions represents the action space.

We show that the above DP is equivalent to a finite action DP (FDP) that has a finite state space for any given starting supply vector a. An important property of an optimal solution of the mVRPSP, which we establish in Theorem 5, serves as the main tool in the construction of the FDP. We also propose a representation of the state space of the FDP by an integer vector, which allows one to reduce the computational work involved in the enumeration of the feasible actions and the state dynamics in the FDP. Additionally, this representation gives rise to the concept of *optimality-invariance* for the mVRPSP, i.e., it allows one to quickly identify all vectors of supply amounts that, for a given set of suppliers, have the same optimal collections of routes. Our FDP formulation can also be easily modified to incorporate many frequently encountered types of constraints on routes (such as pick-up deadlines, etc.).

We solve the FDP by finding a shortest path in the associated directed network. We use A^*, which is a "best-first" search algorithm, for determining a shortest path. When using an appropriate guidance function for A^*, this algorithm (which in its most general form is a heuristic) is known to return an optimal solution. We provide an easily computable guidance function with the desired properties for the FDP formulation of the mVRPSP; thus, our algorithm is an exact algorithm.

The remainder of the paper is organized as follows. In Section 2, we provide a statement of the mVRPSP, as well as two common MIP formulations. Section 3 presents several important properties of optimal solutions of the mVRPSP. In Section 4, we present a DP formulation of the mVRPSP whose state and action spaces are uncountably infinite. We then use results of Section 3 to construct the FDP for given initial conditions that is equivalent to the original DP. The concept of optimality invariance is introduced and used in the reformulation. In Section 5, we discuss how to solve the FDP via a shortest-path search algorithm. In particular, we specify how to implement the A^* algorithm. In Section 6, we illustrate the application of the above shortest path-based algorithm on several examples. In Section 7, we contrast the limitations of the proposed method with its potential computational advantages in practical instances of the mVRPSP, and indicate directions for future research.

2. Preliminaries and the mVRPSP problem statement

We denote the ith unit vector in $\mathbb{R}^d$ by e_i and a vector of all ones by e. For a given vector v, v^{T} stands for its transpose. The non-negative orthant in $\mathbb{R}^d$ is denoted by $\mathbb{R}^d_+$. The set of all d-dimensional vectors of integers is denoted by $\mathbb{Z}^d$, and its intersection with the non-negative orthant by $\mathbb{Z}^d_+$. Finally, for $x \in \mathbb{R}$, $\lceil x \rceil$ is the smallest integer n such that $x \leqslant n$.

Let M be the number of suppliers. We will assume that the suppliers are indexed with integers 1 through M, and the depot is indexed by 0.

Let $a_i \geqslant 0$, $i = 1, \ldots, M$, be the amount of supplies at supplier i, expressed in numbers of truckloads. Let $a = (a_1, \ldots, a_M) \in \mathbb{R}^M_+$ denote the *supply vector*.

Every route originates at the depot (e.g., the assembly plant, in our application context), extends to one or more suppliers, and finally returns to the depot. Each truck picks up parts from the suppliers on the route and makes a delivery at the depot. Each route can be represented by a vector $r \in \{0,1\}^M$, where $r_i = 1$ if and only if the truck following this route visits the ith supplier. Such specification of a route does not describe the order in which the suppliers are visited by the truck. We will, however, assume throughout this paper that the suppliers are visited in the order leading to the minimal cost of the route specified by the vector r (i.e., the travelling salesman tour, perhaps, with some side constraints). The total cost of executing route r is then the sum of the costs of travelling between consecutive suppliers on the route (according to the minimal cost routing), and is denoted by $c(r)$. A route r is called a *dedicated trip* to supplier i if $r = e_i$. There may be certain restrictions in selecting a route in practice, such as "a truck is allowed to visit at most nine suppliers," or "suppliers 1 and 5 may not be on the same route since they are too far apart," and so on. The set of all feasible routes is denoted by R; the set R is finite since the number of suppliers M is finite.

In this paper we consider instances of the mVRPSP in which the incurred costs are determined only by the cost of travel. This would be the case, for example, if the measure of performance to be minimized is the total mileage travelled by the truck fleet, or the total travel time of the trucks. Hence, the cost is independent of load size, and the cost $c(r)$ of a route r is the sum of the costs of traveling between consecutive locations along the route. The cost of travelling from location i to location j is denoted by $c_{ij} \geqslant 0$ for all $i,j \in \{0, \ldots, M\}$ such that $i \neq j$, and assumed to satisfy the triangle inequality. Hence, $r \leqslant r'$ implies $c(r) \leqslant c(r')$.

We assume that the set R satisfies the following conditions:

- $\sum_{r \in R} r \geqslant e$ (i.e., every supplier is visited by at least one feasible route);
- if $r \in R$, then $r' \in R \; \forall r' \in \{0,1\}^M$ such that $r' \leqslant r$ (i.e., if a group of suppliers is allowed to be visited by one truck, any subset of this group can also be visited by one truck).

We remark that many types of real-world constraints readily satisfy the latter condition. For example, including upper bounds on the pick-up times for each supplier or time windows on pick-up times (assuming that, if necessary, the trucks can idle at no additional cost) will produce such collections of routes.

As a truck traverses route r, it picks up loads of supplies at each supplier it visits. We denote the vector of loads picked up by this truck by $w^r \in \mathbb{R}^M_+$. Any feasible vector w^r must satisfy the following two inequalities:

- $e^{\mathrm{T}} w^r \leqslant 1$ (i.e., the capacity of the truck, assumed to be 1 for all trucks in the fleet, cannot be exceeded);
- $w^r \leqslant r$ (i.e., parts can be picked up only from the suppliers that are visited by the truck). In view of this restriction, the first constraint can also be stated as $r^{\mathrm{T}} w^r \leqslant 1$.

We can now formally state the mVRPSP:

Given a set of suppliers $\{1, \ldots, M\}$, a supply vector $a \in \mathbb{R}^M_+$, and a finite set of all feasible routes R with route costs $c(r)$, $r \in R$, find a collection of routes $R^* = (r^1, \ldots, r^F) \in R^F = R \times \cdots \times R$ and the corresponding feasible truck loads $W^* = (w^{r^1}, \ldots, w^{r^F})$ that allow for all supplies to be delivered to the depot at minimal cost.

In the existing literature the mVRPSP has been typically formulated as an MIP, and a large body of research has focused on improving the formulations and developing efficient MIP methods for finding, or obtaining bounds on, their solutions. The bounds were used to judge the quality of heuristic algorithms for the mVRPSP. We provide two of the most basic MIP formulations for completeness. The first formulation arises from a generalization of a wellknown formulation of a multiple vehicle routing problem (with no split pick-ups allowed). References and similar formulations are presented in Lawler et al. (1985). Let U be an upper bound on the number of trucks needed to move all the parts (for example, one might set $U = \sum_{i=1}^{M} \lceil a_i \rceil$). For $k = 1, \ldots, U$ and $i,j = 0, \ldots, M$, let us define the variables as follows:

$$x_{ijk} = \begin{cases} 1, & \text{if truck } k \text{ travels to supplier } j \text{ directly from supplier } i, \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{if supplier } i \text{ is visited by truck } k, \\ 0, & \text{otherwise,} \end{cases}$$

w_{ik} = load picked up at supplier i by vehicle k,

u_{ik} = dummy continuous variables for subtour elimination constraints.

Then the mVRPSP can be formulated as an MIP:

$$\min \quad \sum_i \sum_j c_{ij} \sum_k x_{ijk} \tag{1a}$$

$$\text{s.t.} \quad \sum_j x_{ijk} = \sum_j x_{jik} = y_{ik}, \quad \forall i, \forall k, \tag{1b}$$

$$u_{ik} - u_{jk} + (M+1) \cdot x_{ijk} \leqslant M, \forall i (\neq 0), \quad \forall j (\neq 0), \forall k, \tag{1c}$$

$$w_{ik} \leqslant a_i y_{ik}, \quad \forall i, \forall k, \tag{1d}$$

$$\sum_k w_{ik} = a_i, \quad \forall_i \tag{1e}$$

$$\sum_i w_{ik} \leqslant 1, \quad \forall k, \tag{1f}$$

$$y_{ik}, x_{ijk} \in \{0,1\}, \quad \forall i, \forall j, \forall k, \qquad w_{ik} \geqslant 0, \forall i, \forall k. \tag{1g}$$

The objective function (1a) expresses the total travel cost as a sum of the costs of travelling between consecutive suppliers. Constraints (1b) are flow conservation constraints and serve as definitions of variables y_{ik}. Constraints (1c) are subtour elimination constraints defined for the route of each truck. Constraints (1d) ensure that pick-ups are made only from suppliers visited, while constraints (1e) guarantee that all supplies are picked up. Finally, constraints (1f) ensure that the truck capacity is not exceeded.

The above formulation is rather generic; in particular, it does not include possible constraints on the routes. If any such constraints are present in a problem instance, they would need to be added to the MIP model. This can prove to be a daunting task and complicate the solution methods even further (for example, for an overview of problems with time constrained routes see Desrosiers et al., 1995). The alternative MIP formulation below assumes knowledge and makes explicit use of the collection of feasible routes R and their costs $c(r)$, $r \in R$.

Let U be an upper bound on the number of trucks needed, and let $|R|$ be the cardinality of the set R. For $j = 1, \ldots, U$ and $k = 1, \ldots, |R|$, define the variables as follows:

$$x_{jk} = \begin{cases} 1, & \text{if truck } j \text{ takes route } r^k \\ 0, & \text{otherwise,} \end{cases}$$

$w_j \in \mathbb{R}^M_+$: load vector for truck j.

Then the mVRPSP can be formulated via the following MIP:

$$\min \quad \sum_{k=1}^{|R|} c(r^k) \left(\sum_{j=1}^{J} x_{jk} \right) \tag{2a}$$

$$\text{s.t.} \quad \sum_{k=1}^{|R|} x_{jk} = 1, \quad \forall j \tag{2b}$$

$$w_j \leqslant \sum_{k=1}^{|R|} r^k x_{jk}, \quad \forall j \tag{2c}$$

$$\sum_{j=1}^{J} w_j = a \tag{2d}$$

$$e^{\mathrm{T}} w_j \leqslant 1, \quad \forall j \tag{2e}$$

$$x_{jk} \in \{0,1\}, \quad \forall k, \forall j, \qquad w_j \geqslant 0, \forall j \tag{2f}$$

The objective function (2a) expresses the total travel cost as the sum of the costs of all routes taken by the trucks. Constraints (2b) ensure that each truck is assigned a feasible route (since the "empty" route $r = \vec{0}$ is considered to be feasible, the use of this route will make up for possible over-estimation of the number of trucks necessary). Constraints (2c) ensure that pick-ups are made only from suppliers visited, while constraint (2d) guarantees that all supplies are picked up. Finally, constraints (2e) ensure that the truck capacity is not exceeded.

Formulating the mVRPSP in the form given by (2) requires a pre-processing step, in which the set of feasible routes R is enumerated, and the (minimal) costs $c(r)$ of all routes $r \in R$ are computed. Therefore, the applicability of this formulation relies on the assumption that the computational overhead incurred by including this pre-processing step is not excessive. Since, in practice, the feasible routes can visit only a few locations due to time and distance restrictions, this assumption may be justified in many settings. Moreover, in most applications, different instances of the mVRPSP arise repeatedly and frequently, with the same supplier locations and same constraints on the routes, but different supply amounts. Since the set R is the same for each such instance, the pre-processing step only needs to be performed once, and its results can be re-used when the solution procedure is applied to each instance. A similar formulation was used in Sierksma and Tijssen (1998), combined with a column-generation approach.

The DP approach to the mVRPSP described in this paper, in theory, requires the same pre-processing step. However, as will be apparent from the DP formulation, our algorithm does not need to have access to the entire collection of feasible routes at initialization. Rather, promising routes can be generated and checked for feasibility (and stored for future reference, if desired) in an ad hoc manner while the algorithm is in progress (an idea somewhat similar to a column generation method used in an MIP). Furthermore, since our approach utilizes the notion of optimality invariance, it is especially well-suited to re-solve instances of the mVRPSP with supply amounts varying slightly (in certain cases requiring no additional work, as we will demonstrate).

3. Properties of optimal collections of routes

In this section we explore some properties of optimal collections of routes. For a given set of suppliers and travel costs $c_{ij} \geqslant 0$, let $z : \mathbb{R}_+^M \to \mathbb{R}_+$ denote the function mapping each supply vector $a \in \mathbb{R}_+^M$ to the cost of corresponding optimal routing, $z(a)$. Proposition 1 follows from the assumption that the costs c_{ij} satisfy the triangle inequality.

Proposition 1. *The function $z : \mathbb{R}_+^M \to \mathbb{R}^+$ has the following properties*

i. $z(a) = 0$ *iff* $a = 0$.
ii. *if* $a \geqslant a'$, *then* $z(a) \geqslant z(a')$. (*Isotonicity*)
iii. $z(a + a') \leqslant z(a) + z(a'), \forall a, a' \in \mathbb{R}_+^M$. (*Triangle inequality*)

Let

$$R^* = (r^1, \ldots, r^F)$$

be an optimal collection of routes with respect to the costs c_{ij} for a given supply vector $a \in \mathbb{R}_+^M$. Here, $r^1, \ldots, r^F \in R$, and the routes are allowed to be repeated. In addition, let

$$W^* = (w^{r^1}, \ldots, w^{r^F})$$

be a corresponding optimal collection of (vectors of) loads to be picked up by each route in R^*.

Lemmas 2 and 4 present slight modifications of results of Dror and Trudeau (1990) for the possibly restricted collection of feasible routes R. We say that two routes, r^1 and r^2, have a split supplier i in common if $r_i^1 = r_i^2 = 1$.

Lemma 2. *There exists an optimal collection R^* with no two routes having more than one split supply location in common.*

Proof. Let R^* be an optimal collection of routes. Suppose R^* contains routes r^1 and r^2 that both visit two suppliers, i and $j, i \neq j$, and let $w_i^{r^1}, w_j^{r^1}, w_i^{r^2}$, and $w_j^{r^2}$ be the corresponding loads. Suppose without loss of generality that $w_i^{r^1} = \min\{w_i^{r^1}, w_j^{r^1}, w_i^{r^2}, w_j^{r^2}\}$. Consider the alternative feasible routing decision and the corresponding loads

$$\bar{w}_i^{r^1} = 0, \quad \bar{w}_j^{r^1} = w_j^{r^1} + w_i^{r^1}, \quad \bar{w}_i^{r^2} = w_i^{r^2} + w_i^{r^1}, \quad \bar{w}_j^{r^2} = w_j^{r^2} - w_i^{r^1}.$$

Now R^* can be modified so that r^1 skips supplier i, since $\bar{w}_i^{r^1} = 0$ and omitting a stop on a route results in a feasible route according to our assumptions on R. The new loads are feasible, and the new routing has one fewer split supplier, and is no more expensive than the original routing due to the triangle inequality. Continuing inductively, we can construct an optimal collection of routes with at most one split supplier between any two routes. □

Definition 3. Suppose for a collection of k routes there exist k distinct locations $i_1, \ldots, i_k$ such that route 1 includes locations i_1 and i_2, route 2 includes locations i_2 and i_3, etc. Finally, route k includes locations i_k and i_1. Such collection of locations is called a k-split cycle.

The following lemma generalizes Lemma 2:

Lemma 4. (Dror and Trudeau) *There exists an optimal collection of routes in which there are no k-split cycles, for any* $k \geqslant 2$.

Suppose that $R^* = (r^1, \ldots, r^F)$ is an optimal collection of routes and that the routes are executed one at a time in the order r^1, r^2, r^3, etc. Each route visits one or more suppliers, picking up all or part of the supply remaining at each location at the time of the visit. We will say that the route "empties" a supplier it visits if after the route is executed and a non-zero load is picked up from this location, there is no supply remaining at this location. For the route r^j (i.e., the jth route in the ordering), let

$$I^j = \{i : (r^j)_i = 1, \text{ and supplier } i \text{ is not emptied by } r^j\}. \tag{3}$$

Note that the definition of the sets I^j above depends on the order in which the routes are executed, as well as the loads picked up by each of the trucks. Executing the routes in a different order (or assigning different loads) will lead to different sets I^j. In particular, we will now show that the routes in an optimal collection can be ordered in such a way that every route leaves at most one supplier non-empty, i.e., $|I^j| \leqslant 1, j = 1, \ldots, F$.

Theorem 5. *Suppose R^* contains no k-split cycles for any $k \geqslant 2$. Then the routes in R^* can be ordered to be executed in such a way that $|I^j| \leqslant 1, j = 1, \ldots, F$, where I^j is defined by (3).*

Proof. Let $(w^{r^1}, \ldots, w^{r^F})$ be the loads corresponding to the collection of routes R^*. Let $S^j = \{i : (r^j)_i = 1, w_i^{r^j} < a_i\}$. (This definition is subtly different from I^j. Indeed, S^j consists of indices of suppliers whose *total* supplies exceed the amounts allocated to the route r^j. I^j consists of the indices of the suppliers not emptied by route *j at the time the route is executed*, i.e., the total supply might have been reduced by a previous pickup by the time r^j is executed. As a result, in particular, $I^j \subseteq S^j$). Suppose that $|S^j| \geqslant 2$ for all $j = 1, \ldots, F$. Then, taking into account Lemma 2, every route $r \in R^*$ intersects at least two other routes, i.e., there must be a k-split cycle for some $k \geqslant 2$, contradicting our assumption. Hence, there exists a route $r \in R^*$ with $|S^j| \leqslant 1$.

Let w^r be the load vector corresponding to r. Execute this route first (by construction, it empties all nodes it visits but one) and update the remaining supply amounts and collection of routes: $\hat{a} = a - w^r$, $\hat{R}^* = R^* \setminus r$. It is straightforward to show that $\hat{R}^*$ is an optimal collection of $F - 1$ routes for $\hat{a}$ with no split cycles, and hence the proof can be continued by induction. □

Corollary 6.

(1) *Suppose $a \in \mathbb{Z}_+^M$. Then the optimal collection of routes consists of dedicated trips to every location, at the total cost $z(a) = \sum_{i=1}^M a_i(c_{0i} + c_{i0})$.*

(2) *Suppose* $0 < a_i \leqslant 1$, $i = 1, \ldots, M$ *and* $\lceil \sum_{i=1}^{M} a_i \rceil = \sum_{i=1}^{M} \lceil a_i \rceil = M$. *Then the optimal collection of routes consists of dedicated trips to every location, at the total cost* $z(a) = \sum_{i=1}^{M} (c_{0i} + c_{i0})$.

4. DP formulations of the mVRPSP

We now model the mVRPSP as a dynamic program (DP) having $\mathbb{R}_+^M$ as its state space. We then use the results presented in Section 3 and the notion of optimality invariance to construct an equivalent DP with a finite action space and, for any given initial supply vector, a finite state space.

4.1. A dynamic programming perspective

The mVRPSP can be naturally thought of as a DP as follows: suppose again that trucks are routed sequentially, one at a time, and let decision epochs (or stages) occur when, for each truck in the sequence, the suppliers to be visited by the truck and the loads the truck is to pick up at each of these suppliers are determined. The state of the system at a given stage is described by the vector $a \in \mathbb{R}_+^M$ of supply amounts remaining at the suppliers at the beginning of this epoch. To specify an action with the system in state a, we need to specify the route and the load vector for the next truck in the sequence; as we show below, specifying the load vector is sufficient. For a given load vector $w \in \mathbb{R}_+^M$, let $r(w) \in \{0,1\}^M$ be such that $r_i(w) = 1(=0)$ if and only if $w_i > 0(=0)$, where $i = 1, \ldots, M$. Thus, if a truck is to pick up load vector w, where we assume $e^{\mathrm{T}} w \leqslant 1$ in order to satisfy the truck's capacity constraint, $r(w)$ describes which suppliers must be visited. Then, the set of all possible pick-ups that a single truck can make given the supply vector $a \in \mathbb{R}_+^M$ is

$$W(a) = \{w \in \mathbb{R}_+^M : w \leqslant a, e^{\mathrm{T}} w \leqslant 1, r(w) \in R\}, \tag{4}$$

where R is the collection of all feasible routes. State dynamics are such that if the initial supply vector is a and a truck transports load vector w from the suppliers to the depot, then the next truck encounters supply vector $a - w$. It then follows that the recursive equation for the DP is

$$\begin{aligned} &z(a) = \min_{w \in W(a)} \{c(w) + z(a - w)\}, \quad a \in \mathbb{R}_+^M \setminus \{0\}, \\ &z(0) = 0, \end{aligned} \tag{5}$$

where $c(w) = c(r(w))$ is the cost of the routing decision. We remark that the order in which the actions are executed, i.e., the trucks are dispatched in practice, is inconsequential after the optimal routing and loading decisions have been determined.

The recursive Eq. (5), however, cannot be directly implemented to solve an instance of the mVRPSP due to uncountably infinite state and action spaces. We next show that it is sufficient to consider only a finite number of actions at each stage, leading to a tractable solution approach.

4.2. Action space reduction

The action space of the DP, $W(a)$, can be reduced to a finite set without loss of optimality by restricting actions based on the properties of optimal solutions presented in Section 3. This action space reduction suggests an alternative DP formulation having a finite action space.

Theorem 5 implies that there exists an optimal collection of routes and corresponding load vectors that, at each stage of the DP, empty all but at most one of the suppliers that are visited. Hence at every decision epoch in the above DP we only need to consider routes and corresponding load vectors satisfying this property, removing all but a finite number of actions from $W(a)$.

Let $a \in \mathbb{R}_+^M$ be the current supply vector. Then we can express the decision regarding the route taken by the next truck and the loads picked up by it by the pair (E, I), where $E \subseteq \{1, \ldots, M\}$ is the set of suppliers visited and emptied, and I is the set of suppliers visited, but not emptied, by the truck. Observe that $E \cap I = \emptyset$, either set may be empty, and the cardinality of I is at most one. Given the pair (E, I), it is possible to "reconstruct" the route r taken by the truck by letting

$$r_i(E,I) = \begin{cases} 1, & i \in E \cup I, \\ 0, & \text{otherwise}, \end{cases} \quad i = 1,\ldots,M, \tag{6}$$

and the loads picked up by the truck at each location by letting

$$w_i(E,I) = \begin{cases} a_i, & i \in E, \\ 1 - \sum_{j\in E} a_j, & i \in I, \\ 0, & \text{otherwise} \end{cases} \quad i = 1,\ldots,M. \tag{7}$$

Note that (7) is well-defined since set I contains at most one supplier. To ensure that the action specified by the pair (E,I) leads to a feasible route and load vector, the sets E and I have to be chosen so that $r(E,I) \in R$ and $\sum_{i\in E} w_i = \sum_{i\in E} a_i \leqslant 1$. Moreover, the set I should be non-empty only if $\sum_{j\in E} a_j < 1$, and it is reasonable to restrict the choice of $I = \{i\}$ so that $a_i > 1 - \sum_{j\in E} a_j$ (otherwise supplier i should be included in the set E).

Combining the above observations, the set of feasible actions for the state a can be described as

$$\begin{aligned} A(a) = &\left\{ (E,I) : I = \emptyset, \sum_{j\in E} a_j \leqslant 1, r(E,I) \in R \right\} \\ &\cup \left\{ (E,I) : E \cap I = \emptyset, I = \{i\}, a_i > 1 - \sum_{j\in E} a_j > 0, r(E,I) \in R \right\} \end{aligned} \tag{8}$$

The recursive equation for the DP in this form can be now written as

$$\begin{aligned} z(a) &= \min_{(E,I)\in A(a)} \{c(E,I) + z(a - w(E,I))\}, a \in \mathbb{R}^M_+ \setminus \{0\}, \\ z(0) &= 0, \end{aligned} \tag{9}$$

where $A(a)$ is defined by (8), $w(E,I)$ is determined by (7), and $c(E,I) = c(r(E,I))$, where $r(E,I)$ is defined by (6).

4.3. *n*-vector representation and the finite-state DP formulation

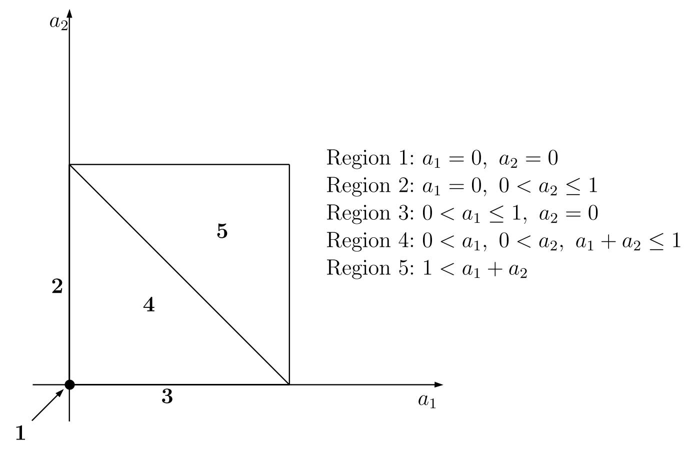
Since we can restrict the set of actions available in each state to a finite collection (8), the number of states encountered by the DP (9) will also be finite for any given $a \in \mathbb{R}^M_+$. We now derive an alternative representation of the state and action spaces, and the recursive equation for this DP, thus representing rigorously the mVRPSP as a DP with finite state and action spaces. The derivation introduces the notion of optimality invariance for the mVRPSP, which is both important as a theoretical concept and has practical ramifications. This representation will also enable us to avoid excessive computation during the execution of the algorithm.

4.3.1. The state space: *n*-vector representation

We begin by presenting an example illustrating the insight that leads to the new state representation. This insight was motivated by our discussions with RIL, as indicated in the introductions and later in this section.

If $M = 2$, then the upper right quadrant of $\mathbb{R}^2$ represents all possible supply vectors (a_1,a_2). If we restrict our attention to the unit square of the quadrant, there are only five different values of $z(a)$:

(1) When $(a_1,a_2) = 0$, no trucks are needed to deliver the supplies, and $z(a) = 0$.
(2) When $a_1 = 0$ and $0 < a_2 \leqslant 1$, a single truck is needed to transport the entire supply from supplier 2 to the depot, and $z(a) = c_{02} + c_{20}$.
(3) When $0 < a_1 \leqslant 1$ and $a_2 = 0$, a single truck is needed to transport the entire supply from supplier 1 to the depot, and $z(a) = c_{01} + c_{10}$.
(4) When $(a_1,a_2) > (0,0)$ but $a_1 + a_2 \leqslant 1$, then both suppliers must be visited but only a single truck is needed. Thus, $z(a) = \min\{c_{01} + c_{12} + c_{20}, c_{02} + c_{21} + c_{10}\} = c(r)$ for $r = (1,1)$, i.e., the cost of a minimum-cost tour that visits both suppliers.
(5) When $(a_1,a_2) > (0,0)$ and $1 < a_1 + a_2 \leqslant 2$, two trucks are needed, one for each supplier. Hence, $z(a) = (c_{01} + c_{10}) + (c_{02} + c_{20})$.



a_2
a_1
Region 1: $a_1 = 0,\ a_2 = 0$
Region 2: $a_1 = 0,\ 0 < a_2 \leq 1$
Region 3: $0 < a_1 \leq 1,\ a_2 = 0$
Region 4: $0 < a_1,\ 0 < a_2,\ a_1 + a_2 \leq 1$
Region 5: $1 < a_1 + a_2$
1
2
3
4
5

Fig. 1. Cost invariant areas within the unit square. Regions are identified in bold.

Thus, the unit square for the $M = 2$ case can be partitioned into five costand solution-invariant areas, Regions 1–5, as depicted in Fig. 1.

Motivated by the above example, we would like to explore whether the set of all possible supply vectors $a \in \mathbb{R}_+^M$ of a general the mVRPSP with M suppliers can be partitioned into such cost- and solution-invariant regions. The rest of this subsection establishes that this is indeed the case. Moreover, for a given supply vector a, the entire solution-invariant region of $\mathbb{R}_+^M$ to which a belongs can be easily characterized.

Let $a \in \mathbb{R}_+^M$ be the supply vector. For an arbitrary subset $J \subseteq \{1, \ldots, M\}$, a lower bound n_J on the number of trucks needed to serve the suppliers in J can be computed as follows:

$$n_J = \left\lceil \sum_{i \in J} a_i \right\rceil \tag{10}$$

(if some of the trucks also serve other suppliers than those in the set J, the actual number of trucks required may be greater than n_J). We denote by $n(a) \in \mathbb{Z}_+^{2\{1,\ldots,M\} \setminus \emptyset}$ the vector of $(2^M - 1)$ such lower bounds for all non-empty subsets of suppliers; we refer to $n(a)$ as the *n-vector* of the state a.

Example 1. Let $a = (1.2, 2.9)$. Then $n(a) \in \mathbb{Z}_+^{\{1\},\{2\},\{1,2\}}$ is defined as follows:

$$n_{\{1\}}(a) = \lceil a_1 \rceil = 2, \quad n_{\{2\}}(a) = \lceil a_2 \rceil = 3, \quad n_{\{1,2\}}(a) = \lceil a_1 + a_2 \rceil = 5.$$

Example 2. Let $a = (1.2, 2.9, 2.2)$. Then

$$\begin{aligned} &n_{\{1\}}(a) = \lceil a_1 \rceil = 2, \quad n_{\{2\}}(a) = \lceil a_2 \rceil = 3, \quad n_{\{3\}}(a) = \lceil a_3 \rceil = 3, \\ &n_{\{1,2\}}(a) = \lceil a_1 + a_2 \rceil = 5, \quad n_{\{1,3\}}(a) = \lceil a_1 + a_3 \rceil = 4, \\ &n_{\{2,3\}}(a) = \lceil a_2 + a_3 \rceil = 6, \quad n_{\{1,2,3\}}(a) = \lceil a_1 + a_2 + a_3 \rceil = 7. \end{aligned}$$

Example 3. Let a^m be any point in region $m, m = 1, \ldots, 5$, in Fig. 1. Then,

$$n(a^1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad n(a^2) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad n(a^3) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad n(a^4) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad n(a^5) = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}.$$

We refer to an arbitrary integer vector $n \in \mathbb{Z}_+^{2^M - 1}$ as *feasible* if there exists $a \in \mathbb{R}_+^M$ such that $n(a) = n$. For a given initial supply vector a, we can restrict our attention to the set of all feasible integer vectors $n \leqslant n(a)$, which is a finite set.

4.3.2. State transitions, action space, and optimality-invariance

When the state a has n-vector $n = n(a)$ and when the selected action is specified by a pair (E, I) as above, the n-vector of the resulting state, $n' = n(a - w(E, I))$, is uniquely determined by $n(a)$ and (E, I). The following theorem formally presents this claim; its proof is constructive.

Theorem 7. *Let $n = n(a)$ be the n-vector of the current state a, and let the chosen action be (E, I). Then the n-vector of the resulting state $n' = n(a - w(E, I))$ is determined by and can be computed using only the information provided by n and (E, I).*

Proof. We will prove the theorem by showing how each component of the vector n' can be computed using only the information provided by n and (E, I). Let $U = \{1, \ldots, M\} \setminus (E \cup I)$ be the set of suppliers not visited by the route $r(E, I)$, and let $J \subseteq \{1, \ldots, M\}$. The following three relationships result from the description of the sets U and E:

- if $J \subseteq E$, then $n'_J = 0$,
- if $J \subseteq U$, then $n'_J = n_J$,
- if $J = \widetilde{E} \cup \widetilde{U}$, where $\widetilde{E} \subseteq E$ and $\widetilde{U} \subseteq U$, then $n'_J = n_{\widetilde{U}}$.

When $I = \emptyset$, the three relationships above completely characterize all components of the vector n'. Suppose $I = \{i_0\}$. Then the following relationships allow us to describe the remaining components of n':

- if $J = I = \{i_0\}$,

$$n'_J = \lceil a_{i0} - w_{i0} \rceil = \left\lceil a_{i0} - \left(1 - \sum_{j \in E} a_j\right)\right\rceil = \left\lceil \sum_{j \in E \cup I} a_j - 1 \right\rceil = n_{E \cup I} - 1,$$

- if $J = \widetilde{E} \cup I$, where $\widetilde{E} \subseteq E$, $n'_J = n'_I = n_{E \cup I} - 1$,
- if $J = \widetilde{U} \cup i_0$, where $\widetilde{U} \subseteq U$,

$$n'_J = \left\lceil \sum_{j \in \widetilde{U}} a_j + a_{i0} - w_{i0} \right\rceil = \left\lceil \sum_{j \in \widetilde{U}} a_j + a_{i0} - \left(1 - \sum_{j \in E} a_j\right)\right\rceil = \left\lceil \sum_{j \in \widetilde{U} \cup I \cup E} a_j - 1 \right\rceil = n_{\widetilde{U} \cup I \cup E} - 1,$$

- if $J = \widetilde{U} \cup \widetilde{E} \cup I$, where $\widetilde{E} \subseteq E$ and $\widetilde{U} \subseteq U$, $n'_J = n'_{\widetilde{U} \cup I} = n_{\widetilde{U} \cup I \cup E} - 1$.

To summarize, the n-vector $n' = n(a - w(E, I))$ of the state resulting from choosing the action (E, I) at this state a is determined by and can be computed using only the information provided by $n(a)$ and (E, I) as follows:

$$n'(n, (E, I))_J = \begin{cases} n_{J \cup E} - 1, & \text{if } I \neq \emptyset \text{ and } I \subseteq J, \\ n_{J \setminus E}, & \text{otherwise}, \end{cases} \quad J \in 2^{\{1, \ldots, M\}} \setminus \emptyset. \qquad \square \tag{11}$$

In view of Theorem 7, we can use the n-vectors as the states of our DP, in place of the supply vectors a. The finite action space for a given state n can be defined as follows:

$$A(n) = \{(E, I) : I = \emptyset, n_E = 1, r(E, I) \in R\} \cup \{(E, I) : |I| = 1, n_E = 1, n_{E \cup I} > 1, r(E, I) \in R\}. \tag{12}$$

We refer to the state n' as a *successor* of state n if and only if the state n' can be attained by executing a feasible action at state n. The DP presented in (9) now can be rewritten based on the n-vector representation of the states as follows:

$$\begin{aligned} &z(n) = \min_{(E,I) \in A(n)} \{c(E, I) + z(n'(n, (E, I)))\}, \qquad n \in \mathbb{Z}_+^{2\{1, \ldots, M\} \setminus \emptyset}, \quad n \neq 0 \\ &z(0) = 0, \end{aligned} \tag{13}$$

where the successor state $n'(n,(E,I))$ is obtained by (11). In the DP (13), the state is represented by an integer vector and the action is represented by a pair of subsets of the supplier set.

An interesting consequence of Theorem 7 is that for given n-vector n, the optimal cost as well as the optimal collection of routes are invariant for all initial supply vectors in the region $\{a \in \mathbb{R}_+^M : n(a) = n\}$ (although the particular pickup amounts may vary). We refer to this property as *optimality invariance*. A practical implication of optimality invariance, supporting a common industry practice, is the observation that the routing decisions do not need to be changed when the supply amounts vary slightly. It should also be noted, however, that when the number suppliers is large, the size of the invariant region is quite small. In particular, even a small decrease in the supply amounts can lead to savings in transportation costs; these savings will not materialize if the above "rule of thumb" is applied without checking that the optimality invariance condition is satisfied. Our characterization of optimality-invariance through the n-vectors provides a quick and simple way to check if the routes need to be changed.

Also note that representing the state of the DP by its n-vector allows us to avoid the computational effort of searching for all feasible actions by the formulas in (8). Instead, checking feasibility is accomplished by a simple lookup of the appropriate component of the n-vector, and the updates of the n-vectors using (11) are also performed without excessive computation.

Finally, notice that although we have assumed thus far that the collection of all feasible routes R (together with the cost of each route) is explicitly specified prior to the execution of the algorithm, Eq. (12) can be used to generate promising routes, and check their costs, in an ad hoc manner while solving the DP. In particular, to enumerate feasible actions in the set $A(n)$, one could pick an action (E,I) that satisfies the conditions of (12) on E, I and n, and then a subroutine can be called to check if the resulting route $r(E,I)$ satisfies all the constraints imposed on the routes, and to evaluate its cost. This approach avoids enumerating and costing-out the routes that are not useful for the instance at hand, and will only produce a relevant subset of R.

5. Shortest path search algorithm

In the previous section we succeeded in representing the mVRPSP as a deterministic DP with finitely many states at each stage, and finitely many actions available at each state. As is common, we will associate this DP with a graph in which each node corresponds to a DP state, directed arcs correspond to transitions from each state to its successor states, and each arc has a cost of the action resulting in this transition associated with it (see, for example, Bertsekas (1995)). Solving the DP is then equivalent to finding a shortest path in this network, from node $n(a)$ to node $\vec{0}$, where a is the initial supply vector. In this section we motivate and describe the particular algorithm we chose to search for the shortest path in the network associated with the DP defined by (13).

We chose a forward-search shortest path algorithm (as opposed to a backwardsearch DP algorithm), since this approach allows us to avoid considering states that are not reachable from the initial state $n(a)$. There are many versions of the basic forward-search shortest path algorithm, distinguished by the order in which they examine the nodes of the network, e.g., depth-first, breadth-first, etc. We opted for the A^* heuristic algorithm, which is a "best-first" forwardsearch shortest path algorithm. More specifically, A^* uses a guidance function for each node to pick the node to be explored next. It is known that A^* returns an optimal solution to the shortest path problem if at each node the guidance function incorporates a lower bound on the true cost of the optimal path from this node to the destination. Efficiency of the implementation of A^* depends in part on the ease of obtaining tight lower bounds. In our setting, lower bounds can be readily obtained, making A^* an appropriate choice.

5.1. The A^* algorithm

The A^* algorithm is presented and analyzed in Pearl (1984) and elsewhere. In its most general form A^* is a forward-search heuristic that seeks a shortest path from a starting node s to an end node t in a given network. Throughout the search, A^* generates nodes of the network as successors (or descendants) of previously generated nodes; pointers from the successor nodes to their parent nodes are maintained. We say that a node has been "explored" if A^* has generated at least one of its successors; we say that a node has been "expanded" if all of its successors have been generated. We refer to the set of all nodes that have been generated but not yet explored as "OPEN" and the set of nodes that have been expanded as "CLOSED."

The algorithm utilizes a guidance function $f(\cdot)$ to choose which of the generated nodes to explore at the next step of the search and thus to guide the search to the most promising alternative in order to find a good solution in its early stage. The guidance function for node n is defined as

$$f(n) = g(n) + h(n),$$

where $g(n)$ is the cost of the best currently known path from the start node s to the node n, with $g(s) = 0$, and $h(n)$ is an estimate of the cost of the optimal path from the candidate node n to the end node t ("cost to go"), with $h(t) = 0$.

The general outline of A^* is as follows (see Pearl, 1984, for details):

(1) Put the start node s into OPEN; set $g(s) = 0$.
(2) If OPEN is empty, exit with failure.
(3) Remove from OPEN a node n for which f is minimized, and place it in CLOSED.
(4) If $n = t$, exit successfully with the solution obtained by tracing back the pointers from n to s
(5) Otherwise expand n, generating all its successors, and attach to them pointers back to n. For every successor n' of n:
   (a) If n' is not already in OPEN or CLOSED, compute the estimate $h(n')$, and calculate $f(n') = g(n') + h(n')$ with $g(n') = g(n) + c(n,n')$, where $c(n,n')$ is the cost of the arc from n to n'.
   (b) If n' is already in OPEN or CLOSED, direct its pointers along the path yielding the lowest $g(n')$.
   (c) If n' required pointer adjustment and was found in CLOSED, reopen it.
(6) Go to step 2.

5.2. Lower bounds

The A^* algorithm is guaranteed to terminate with an optimal solution to the shortest path problem when the estimate function $h(n)$ is a lower bound on the true cost of the optimal path from the node n to the destination (Pearl, 1984). Below we present two lower bounds and discuss their numerical implications.

Lemma 8. $z(a) \geqslant \sum_{i=1}^{M} a_i \cdot (c_{0i} + c_{i0})$.

Proof. First suppose $a \in \mathbb{Q}_+^M$, the set of rational numbers in $\mathbb{R}_+^M$. Then there exists $\alpha \in \mathbb{Z}_+$ such that $\alpha \cdot a \in \mathbb{Z}_+^M$. By Proposition 1, $z(\alpha \cdot a) \leqslant \alpha \cdot z(a)$. Then, using Corollary 6, $z(a) \geqslant \frac{1}{\alpha} \cdot z(\alpha \cdot a) = \frac{1}{\alpha} \sum_{i=1}^{M} \alpha \cdot a_i \cdot (c_{0i} + c_{i0}) = \sum_{i=1}^{M} a_i \cdot (c_{0i} + c_{i0})$.

For any $a \in \mathbb{R}_+^M$, there exists a sequence of vectors $\{a^k\}_{k=1}^{\infty} \subset \mathbb{Q}_+^M$ such that $n(a^k) = n(a)$ for all k. By the optimality invariance property,

$$z(a) = z(a^k) \geqslant \sum_{i=1}^{M} a_i^k \cdot (c_{0i} + c_{i0}) \to \sum_{i=1}^{M} a_i \cdot (c_{0i} + c_{i0}) \quad \text{as } k \to \infty. \qquad \square$$

As an example of the bound provided by Lemma 8, if two suppliers 1 and 2 have supplies of 1.3 and 0.9 respectively, a lower bound can be computed as the sum of 1.3 times the cost for the dedicated trip to the supplier 1 and 0.9 times the cost for the dedicated trip to the supplier 2.

Another lower bound can be defined as follows:

$$\sum_{i=1}^{M} (c_{0i} + c_{i0}) \cdot \lfloor a_i \rfloor = \sum_{i=1}^{M} (c_{0i} + c_{i0}) \cdot \max(0, \lceil a_i \rceil - 1). \tag{14}$$

Clearly, the lower bound presented in Lemma 8 is tighter than the lower bound presented in Eq. (14). It is shown in Pearl (1984) that strictly tighter bounds will not degrade algorithmic performance, and numerical experience indicates that, other things being equal, tighter lower bounds usually improve algorithmic performance. However in order to compute the first lower bound, we need to know the exact supply vector a. Recall that in the DP formulation of the mVRPSP (13) we discarded this information (in part to avoid excessive computation in determining the set of feasible actions and the successor state), and stored the vector $n(a)$ instead. The second bound (14) can be computed by using the appropriate components of the vector $n(a)$ only. Computational experience suggests that overall performance of the search is improved by using a loose but simple lower bound (14) rather than the tight but computationally demanding lower bound of Lemma 8.

5.3. Backtracking the path

A^* terminates successfully when it reaches the end node $\vec{0}$. The shortest path from node $n(a)$ to node $\vec{0}$ can be obtained by "backtracking" the pointers from successor nodes to their parents, starting from node $\vec{0}$.

Once the shortest path from $n(a)$ to $\vec{0}$ has been established, we interpret the arcs of the network along the path as the actions in the DP as they provide the routes and load vectors for the trucks. Each arc of the path constructed by A^* corresponds to an action defined by a pair of sets (E, I), as described in Section 4.2. The action corresponding to an arc from node n to node n' of the network can be either stored throughout the algorithm along with pointer information, or, alternately, can be derived by comparing the appropriate components of the vectors n and n'. Given the pair (E, I), the route of the truck and the load vector are specified by Eqs. (6) and (7).

6. Numerical examples

So far we have tested our algorithm on smaller instances of problems than ones typically encounters in practice, with the primary goal of illustrating and testing the concept. We summarize the results of our numerical examples below.

We generated instances of the mVRPSP by specifying the geographic layout and the supply vectors. To consider different geographic layouts, we located the depot at the origin, and generated the (x, y) coordinates of the suppliers randomly. We thus generated nine geographic layouts N4L1, N4L2, N4L3, N5L1, N5L2, N5L3, N7L1, N7L2, and N7L3 described in Table A.1. The first number in the code name of the layout refers to the number of suppliers considered. The distances were assumed to be Euclidean, and we assumed that all possible routes were feasible. The supply vectors, specified in Table A.2, were generated randomly to yield, for every layout, problem instances with total supply gradually increasing from 1.2 up to 9.6, with an incremental step of 0.4. The nine layouts listed in Table A.1 were paired with the 22 corresponding supply vectors listed in Table A.2 to create 198 problem instances.

We implemented the shortest path approach for solving the DP (13) on a laptop computer with an Intel microprocessor Pentium M 1.6 GHz using C++. Our implementation includes a subroutine that efficiently enumerates feasible actions using (12). We did not, however, take advantage of the possibility to generate routes in an ad hoc manner mentioned in Section 4; instead, since we intended to consider 22 different supply vectors for each layout, we computed the cost of the routes in a separate pre-processing step. Since the problem instances were relatively small, the pre-processing step took negligible time. We used the lower bound (14) in the implementation of A^*.

Incidentally, we also attempted to solve MIP formulations (1) and (2) for each instance, using AMPL with CPLEX 9.0 on the same personal computer. We used the default settings of the CPLEX MIP solver, and made no attempt to exploit the special structure of the MIPs with specialized solution approaches, since solution procedures for the MIPs have not been the focus of our research. Specialized solutions approaches, such as the heuristics and approximate algorithms presented in the papers discussed in Section 1, can certainly be used to reduce the MIP solution time. It is, however, remarkable that the need for the specialized MIP approaches arises even for the small problem instances we considered here, which the simple basic implementation of the shortest-path based approach was able to solve. We note that formulation (2) is, in fact, a set covering type formulation in which travelling salesman problems for every possible subset of supplier sets are solved a priori and used as a part of the formulation. Therefore, MIP (2) utilizes additional information and hence shows, as expected, superior performance, relative to MIP (1).

The computation times in seconds of MIP (1), MIP (2) and the shortest path approach (SPA) for each instance, are listed in Table A.3 for $M = 4$ and in Table A.4 for $M = 5$. Note that the shortest path approach solved each of these instances in under a second, while the MIP approaches took significantly longer on all but the simplest instances. As indicated in the tables, in several instances of each of the layouts CPLEX failed to converge after an hour of CPU time has elapsed. Instances with $M = 7$ were too large to be solved by the MIP approaches. Hence, only the computation times of the shortest path approach for these problems are reported in Table A.5.

We observe that, as a general trend, the computational effort required for each of the approaches increases as the total supply increases and that this increase is dramatic for larger values of the total supply. It is also interesting to observe that the runtime does not increase monotonically in the total supply. Certain patterns of allocating supplies among suppliers seem to make the problem more demanding than other patterns in terms of the computation time required.

Typical output of the algorithm presented below for examples with nine suppliers. The locations of supplies are given in the second row of Table1, and the supply amounts were varied to arrive at three examples.

It took 2737 s, or about 45 min, to solve the first example problem. The optimal solution found is as follows:

$r^1 = (1,0,0,0,0,0,1,1,0)$, or $8(0.2) \mapsto 7(0.5) \mapsto 1(0.3)$,	$c(r^1) = 28.2250$,
$r^2 = (0,0,0,0,1,0,0,0,0)$, or $5(1.0)$,	$c(r^2) = 14.1421$,
$r^3 = (0,0,1,0,0,0,0,0,0)$, or $3(1.0)$,	$c(r^3) = 18.8680$,
$r^4 = (0,1,0,0,1,1,0,0,0)$, or $6(0.3) \mapsto 5(0.2) \mapsto 2(0.5)$,	$c(r^4) = 14.9020$,
$r^5 = (0,0,1,0,0,0,0,0,1)$, or $9(0.3) \mapsto 3(0.3)$,	$c(r^5) = 19.8764$,
$r^6 = (0,0,0,1,0,1,0,0,0)$, or $6(0.5) \mapsto 4(0.5)$,	$c(r^6) = 12.8371$.

The loads are given in parentheses, and the total cost is 108.8506 units. Note that supplier 6 has multiple visits by r^4 and r^6, although the supply at this location is less than truck capacity. Split pick-ups at suppliers 3 and 5 are unavoidable because their supplies exceed the truck capacity.

In the next two examples we used the same supplier layout as above, altering supply amounts as indicated in Table 1. The examples demonstrate the savings that can be realized by split pick-ups (for comparison, the cost of nine dedicated trips is 132.866 units). Although it took the algorithm several hours to solve these example problems, the intricate nature of optimal collections of routes, especially in Example3, illuminates the difficulty of solving such problems to optimality. The optimal solution for Example 2 found by the algorithm is as follows:

r^1: 9(0.92),	$c(r^1) = 14.5602$,
r^2: 4(0.53),	$c(r^2) = 7.2111$,
r^3: 3(0.88),	$c(r^3) = 18.8680$,
r^4: 1(0.77),	$c(r^4) = 8.2462$,
r^5: 6(0.80),	$c(r^5) = 5.6569$,
r^6: 5(0.95),	$c(r^6) = 14.1421$,
r^7: $7(0.74) \mapsto 8(0.26)$,	$c(r^7) = 27.2599$,
r^8: $8(0.27) \mapsto 2(0.63)$,	$c(r^8) = 31.9789$,

The total cost of executing these routes is 127.923 units.

Example3 differs from Example2 by supply amounts at two of the suppliers. For this example, the algorithm produced the following optimal solution:

Table 1
Layout and supply amounts of the suppliers for $M = 9$ examples

Supplier	1	2	3	4	5	6	7	8	9
(x, y)	$(4, -1)$	$(5, 3)$	$(-8, 5)$	$(-3, -2)$	$(5, 5)$	$(2, 2)$	$(9, -10)$	$(8, -10)$	$(-7, 2)$
Example1	0.30	0.50	1.30	0.50	1.20	0.80	0.50	0.20	0.30
Example2	0.77	0.63	0.88	0.53	0.95	0.80	0.74	0.53	0.92
Example 3	0.51	0.63	0.58	0.93	0.95	0.80	0.74	0.53	0.92

r^1: 9(0.92),	$c(r^1) = 14.5602$,
r^2: 4(0.93),	$c(r^2) = 7.2111$,
r^3: 3(0.58) $\mapsto$ 6(0.42),	$c(r^3) = 22.7027$,
r^4: 6(0.01) $\mapsto$ 5(0.95),	$c(r^4) = 14.1421$,
r^5: 2(0.63) $\mapsto$ 6(0.37),	$c(r^5) = 11.8217$,
r^6: 8(0.26) $\mapsto$ 7(0.74),	$c(r^6) = 27.2599$,
r^7: 8(0.27) $\mapsto$ 1(0.51),	$c(r^7) = 26.7782$,

with a total cost of 124.476 units. Notice the complicated structure of the optimal collection of routes: supplier 6 is visited by three trucks, and supplier 8—by two trucks. Such combinations of routes would be difficult to identify by relying on a heuristic or by having a dispatcher solving the problem by hand.

7. Discussion and future research

The main contributions of this paper are:

- Based on insights from third-party logistics professionals, we have identified and described an optimality-invariance condition for the mVRPSP.
- We have used the optimality-invariance condition to transform a computationally intractable DP formulation of the mVRPSP, having uncountable state and action spaces, into a potentially tractable DP with finite state and action spaces.

These contributions represent a new direction for determining optimal solutions for the mVRPSP.

Additionally, we cast the finite state and action DP as a problem of finding a minimum-cost path in a finite, directed graph. We then used A^* as a solution technique. A numerical analysis indicated that this particular approach for solving the mVRPSP shows significant promise, although like other DP-based approaches, it can suffer from the "curse of dimensionality."

Future research directions include finding better solution procedures for solving the minimum-cost path problem, considering ways other than the minimum cost path problem for solving the finite state and action DP, and examining heuristic solution procedures. All of these directions focus on determining optimal or near-optimal solutions to real-world instances of the mVRPSP, i.e., instances with $M \geqslant 50$. With regard to finding better solution procedures, the current straightforward implementation of A^* could be improved by incorporating computation saving techniques, such as ad hoc route generation procedures, and by adding a pruning step to take advantage of a feasible solution provided by an experienced dispatcher (or heuristic) that would leave unpromising paths unexplored.

Computationally desirable heuristic solution procedures are likely to be based on insights provided by practitioners. For example, according to RIL, most truck routes are limited to at most 12 stops, and a typical route rarely involves more than six stops in some applications; e.g., delivering from automotive suppliers to an assembly plant. Thus, it is reasonable in a variety of circumstances to limit the number of stops that a single route can serve, reducing the size of the action space in the DP and hence the number of reachable states. Also, it is often the case that suppliers are clustered geographically, and in such cases, it may be reasonable to only consider routes that visit suppliers within one cluster, decomposing the problem into smaller, more manageable sub-problems.

Another topic of future research is the incorporation of even more general constraints on the routes. As we mentioned earlier, adding constraints on the routes to the MIP formulation of the mVRPSP can be extremely difficult, requiring the addition of many binary variables. In our framework, any collection of constraints on the routes can be easily incorporated, especially if the routes are generated in an ad hoc manner as described in Section 4. The only restriction governing the types of constraints that can be incorporated is that the resulting collection of feasible routes must satisfy the assumptions of Section 2; namely, that any portion of a feasible route is itself feasible. Many types of constraints encountered in practice satisfy the above assumption. For example, including upper bounds on the pick-up times for each supplier, or time windows on the pick-up times (assuming that, if necessary, the trucks can idle at no additional cost) will produce such collections of routes.

Applicability of the DP-based approach to such problems is conceptually simple. In practice, one should consider whether the feasible routes satisfying the constraints arising in a specific problem can be efficiently generated in a pre-processing step, or in an ad hoc manner, or with a combined approach. It is also a potentially interesting problem to consider how the above assumption can be checked for a given set of constraints without explicitly enumerating all feasible routes. An even broader area of further research is to expand the concept of optimality-invariance, and utilize it in algorithm design, for more general classes of the mVRPSP and other problems.

Acknowledgements

The authors express their appreciation to the Alfred P. Sloan Foundation, the University of Michigan Trucking Industry Program (UMTIP), Ryder Integrated Logistics, and the Department of Industrial and Operations Engineering at the University of Michigan for supporting the research reported in this paper. UMTIP is part of the Sloan Industry Centers Network.

Appendix A. Data for the numerical examples

See Tables A.1–A.5

Table A.1
Geographic layouts for the problem instances

Code	M	Position						
N4L1	4	1(1, −3)	2(−6, −3)	3(−2, −8)	4(0, −7)			
N4L2	4	1(7, 7)	2(−2, 0)	3(3, 8)	4(−9, 1)			
N4L3	4	1(1, −4)	2(3, 1)	3(2, 6)	4(8, −1)			
N5L1	5	1(2, 7)	2(9, 2)	3(9, −7)	4(−1, −7)	5(8, −7)		
N5L2	5	1(−10, −6)	2(−10, 0)	3(−4, 7)	4(1, 1)	5(3, −10)		
N5L3	5	1(4, −8)	2(−2, 5)	3(2, −6)	4(−4, −3)	5(1, 2)		
N7L1	7	1(4, −6)	2(2, 6)	3(7, 7)	4(5, −5)	5(4, 9)	6(−8, 0)	7(5, −7)
N7L2	7	1(−10, −6)	2(−10, 0)	3(−4, 7)	4(1, 1)	5(3, −10)	6(9, −10)	7(−1, 4)
N7L3	7	1(4, −8)	2(−2, 5)	3(2, −6)	4(−4, −3)	5(1, 2)	6(6, −3)	7(−1, 0)

Table A.2
Supply vectors

Code	$M=4$				$M=5$					$M=7$						
	a_1	a_2	a_3	a_4	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5	a_6	a_7
Q1	0.55	0.40	0.24	0.01	0.02	0.14	0.56	0.23	0.25	0.26	0.07	0.01	0.01	0.22	0.31	0.32
Q2	0.19	0.76	0.31	0.35	1.01	0.46	0.12	0.01	0.01	0.33	0.34	0.09	0.37	0.25	0.19	0.03
Q3	1.27	0.57	0.15	0.01	0.28	0.40	0.42	0.45	0.45	0.26	0.34	0.35	0.23	0.13	0.38	0.31
Q4	0.01	0.61	0.86	0.92	0.24	0.94	0.64	0.50	0.08	0.56	0.54	0.31	0.08	0.27	0.14	0.50
Q5	0.83	0.83	0.23	0.91	0.56	0.73	0.75	0.48	0.28	0.12	0.45	0.49	0.58	0.58	0.35	0.23
Q6	0.98	0.77	0.12	1.32	0.70	0.58	0.76	0.74	0.43	0.33	0.37	1.04	0.03	0.47	0.12	0.84
Q7	1.17	1.20	0.78	0.45	0.27	0.87	0.44	1.62	0.39	0.07	0.01	1.18	0.35	0.35	0.75	0.88
Q8	1.01	0.83	1.10	1.06	0.74	0.80	0.94	0.95	0.58	0.85	0.74	0.49	0.21	0.76	0.48	0.47
Q9	1.72	0.45	1.47	0.75	0.95	0.64	0.72	2.03	0.06	1.01	0.79	0.12	0.64	0.41	0.78	0.65
Q10	1.54	0.37	1.39	1.50	1.49	0.37	2.68	0.22	0.03	0.90	0.57	0.24	0.35	0.67	1.26	0.81
Q11	1.73	1.73	1.06	0.68	1.74	0.52	0.52	1.11	1.31	1.48	1.13	0.52	0.25	0.99	0.74	0.10
Q12	1.04	1.17	3.30	0.09	1.56	1.36	0.91	0.38	1.39	0.97	0.70	0.20	1.32	0.20	1.25	0.96
Q13	1.88	0.46	3.38	0.28	1.03	1.01	2.09	1.64	0.24	0.52	0.74	0.12	1.67	1.18	0.26	1.51
Q14	0.04	3.98	1.20	1.18	1.32	0.85	1.62	1.34	1.26	1.63	1.16	0.38	0.35	1.62	0.94	0.31
Q15	1.41	1.65	2.00	1.74	1.25	0.52	0.78	1.47	2.78	0.87	1.27	0.70	0.48	0.98	1.18	1.31
Q16	1.84	0.78	2.81	1.77	1.34	2.57	1.95	0.90	0.44	1.69	1.11	0.29	0.86	1.95	0.08	1.21
Q17	1.54	3.19	2.50	0.36	2.07	1.55	0.20	2.20	1.59	1.27	0.82	0.20	1.82	1.77	1.05	0.66
Q18	2.06	1.32	2.53	2.09	0.42	2.68	0.40	2.55	1.95	1.01	0.81	1.82	0.68	0.98	0.58	2.11
Q19	3.67	2.32	0.97	1.44	1.04	1.46	0.24	3.31	2.34	0.66	1.63	1.25	0.43	1.68	1.86	0.88
Q20	1.37	2.58	1.66	3.19	0.51	2.94	2.75	1.95	0.64	1.24	2.20	0.07	2.45	1.41	0.12	1.31
Q21	3.59	1.65	0.81	3.14	0.77	3.53	2.05	0.67	2.18	1.11	0.34	2.35	1.89	0.62	2.33	0.56
Q22	2.68	0.35	3.82	2.75	2.63	1.46	1.01	2.05	2.45	1.84	2.53	0.40	1.48	1.89	0.54	0.92

Table A.3
CPU time and cost for $M=4$

Code($\sum a_i$)	N4L1				N4L2				N4L3			
	Cost	SPA	MIP(1)	MIP(2)	Cost	SPA	MIP(1)	MIP(2)	Cost	SPA	MIP(1)	MIP(2)
Q1(1.20)	28.67	< 1.00	1	< 1.00	40.69	< 1.00	< 1.00	< 1.00	32.34	< 1.00	< 1.00	< 1.00
Q2(1.61)	31.18	< 1.00	2	< 1.00	40.97	< 1.00	< 1.00	< 1.00	33.61	< 1.00	< 1.00	< 1.00
Q3(2.00)	28.96	< 1.00	< 1.00	< 1.00	56.79	< 1.00	1.51	< 1.00	36.79	< 1.00	2	< 1.00
Q4(2.40)	44.19	< 1.00	21	< 1.00	44.68	< 1.00	<< 1.00	< 1.00	44.68	< 1.00	3	< 1.00
Q5(2.80)	44.73	< 1.00	34	1	59.00	< 1.00	5	1	43.34	< 1.00	3	< 1.00
Q6(3.19)	51.22	< 1.00	159	5	73.40	< 1.00	86	12	54.30	< 1.00	91	< 1.00
Q7(3.60)	55.38	< 1.00	313	14	64.49	< 1.00	9	< 1.00	47.51	< 1.00	40	12
Q8(4.00)	59.45	< 1.00	755	20	91.98	< 1.00	872	113	64.79	< 1.00	561	90
Q9(4.39)	64.50	< 1.00	1339	24	94.77	< 1.00	454	58	59.85	< 1.00	74	3
Q10(4.80)	71.17	< 1.00	>1 h	62	95.69	< 1.00	124	16	73.30	< 1.00	2111	36
Q11(5.20)	73.46	< 1.00	1245	58	85.56	< 1.00	95	6	65.40	< 1.00	400	28
Q12(5.60)	91.85	< 1.00	>1 h	351	112.05	< 1.00	101	139	81.07	< 1.00	1219	83
Q13(6.00)	90.44	< 1.00	>1 h	77	125.14	< 1.00	873	31	82.96	< 1.00	95	9
Q14(6.40)	101.93	< 1.00	>1 h	36	88.17	< 1.00	2846	15	81.35	< 1.00	220	43
Q15(6.80)	97.88	< 1.00	>1 h	23	117.51	< 1.00	864	32	85.27	< 1.00	1594	5
Q16(7.20)	103.54	< 1.00	>1 h	304	131.08	< 1.00	2843	115	93.01	< 1.00	988	59
Q17(7.60)	113.91	< 1.00	>1 h	333	120.05	< 1.00	2040	463	86.91	< 1.00	788	79
Q18(8.00)	109.68	< 1.00	>1 h	1292	150.98	< 1.00	>1 h	>1 h	108.13	< 1.00	>1 h	>1 h
Q19(8.40)	103.54	< 1.00	>1 h	1147	140.52	< 1.00	>1 h	674	91.02	< 1.00	2798	160
Q20(8.80)	126.00	< 1.00	>1 h	2014	153.29	< 1.00	>1 h	>1 h	116.79	< 1.00	>1 h	2812
Q21(9.20)	111.61	< 1.00	>1 h	293	172.74	< 1.00	>1 h	>1 h	116.64	< 1.00	>1 h	>1 h
Q22(9.60)	134.85	< 1.00	>1 h	>1 h	186.08	< 1.00	>1 h	158	130.03	< 1.00	>1 h	772

Table A.4
CPU time and cost for $M = 5$

Code($\sum a_i$)	N5L1				N5L2				N5L3			
	Cost	SPA	MIP (1)	MIP (2)	Cost	SPA	MIP (1)	MIP (2)	Cost	SPA	MIP (1)	MIP (2)
Q1(1.20)	50.65	< 1.00	2	< 1.00	50.15	< 1.00	< 1.00	< 1.00	36.58	< 1.00	< 1.00	< 1.00
Q2(1.61)	56.52	< 1.00	2	< 1.00	71.81	< 1.00	4	< 1.00	51.09	< 1.00	9	< 1.00
Q3(2.00)	62.28	< 1.00	14	< 1.00	65.81	< 1.00	7	< 1.00	39.96	< 1.00	3	< 1.00
Q4(2.40)	67.14	< 1.00	39	< 1.00	69.25	< 1.00	3	< 1.00	42.60	< 1.00	4	< 1.00
Q5(2.80)	82.48	< 1.00	271	3	74.66	< 1.00	15	3	51.37	< 1.00	41	3
Q6(3.19)	80.21	< 1.00	242	< 1.00	82.32	< 1.00	81	< 1.00	55.49	< 1.00	91	4
Q7(3.60)	83.24	< 1.00	117	< 1.00	72.98	< 1.00	8	< 1.00	53.34	< 1.00	25	1
Q8(4.01)	91.21	< 1.00	1068	3	83.16	< 1.00	30	3	55.78	< 1.00	35	< 1.00
Q9(4.40)	89.76	< 1.00	143	< 1.00	83.11	< 1.00	54	< 1.00	67.79	< 1.00	45	4
Q10(4.79)	113.74	< 1.00	1459	3	112.90	< 1.00	251	3	89.04	< 1.00	197	18
Q11(5.20)	111.43	< 1.00	2497	10	124.02	< 1.00	1233	10	80.15	< 1.00	1870	114
Q12(5.60)	128.87	< 1.00	>1 h	34	131.03	< 1.00	703	34	81.02	< 1.00	1054	36
Q13(6.00)	142.60	< 1.00	>1hr	>1 h	128.55	< 1.00	>1 h	>1 h	99.30	< 1.00	>1 h	>1 h
Q14(6.40)	142.91	< 1.00	>1 h	1137	136.15	< 1.00	>1 h	1137	88.18	< 1.00	>1 h	575
Q15(6.80)	154.53	< 1.00	>1 h	408	135.41	< 1.00	3296	408	88.10	< 1.00	3452	575
Q16(7.20)	154.88	< 1.00	>1 h	1294	146.94	< 1.00	>1 h	1294	107.12	< 1.00	>1 h	>1 h
Q17(7.60)	150.68	< 1.00	>1 h	864	151.16	< 1.00	>1 h	864	109.74	< 1.00	>1 h	>1 h
Q18(8.00)	168.08	< 1.00	>1 h	>1 h	144.29	< 1.00	>1 h	>1 h	89.35	< 1.00	>1 h	332
Q19(8.40)	171.52	< 1.00	>1 h	>1 h	150.90	< 1.00	>1 h	>1 h	102.95	< 1.00	>1 h	>1 h
Q20(8.80)	185.38	< 1.00	>1 h	1270	158.23	< 1.00	>1 h	1270	112.33	< 1.00	>1 h	551
Q21(9.20)	202.00	< 1.00	>1 h	>1 h	199.82	< 1.00	>1 h	>1 h	106.51	< 1.00	>1 h	>1 h
Q22(9.60)	191.02	< 1.00	>1 h	>1 h	198.80	< 1.00	>1 h	>1 h	113.49	< 1.00	>1 h	>1 h

Table A.5
Problem costs and SPA runtime in seconds ($M = 7$ case)

Code	N7L1		N7L2		N7L3	
	Cost	Time	Cost	Time	Cost	Time
Q1	64.0861	< 1.00	80.8609	< 1.00	54.7519	< 1.00
Q2	64.0861	< 1.00	80.8609	< 1.00	54.7519	< 1.00
Q3	52.3265	< 1.00	65.7453	< 1.00	38.3549	< 1.00
Q4	66.8505	< 1.00	67.9696	1	40.4939	1
Q5	57.1315	1	74.6459	1	42.9713	< 1.00
Q6	77.2731	2	81.4072	2	48.8880	2
Q7	74.7172	14	76.8123	3	46.7315	2
Q8	88.6671	3	90.1050	3	53.143	3
Q9	85.7989	4	99.7638	4	55.6204	4
Q10	89.1976	6	109.1017	6	61.0248	6
Q11	93.4587	9	112.5159	8	66.2124	8
Q12	107.6015	20	116.8531	19	71.9638	20
Q13	101.7885	10	136.0982	10	83.5187	10
Q14	120.2602	9	120.0435	10	78.5852	10
Q15	128.5039	37	115.0287	38	61.5824	33
Q16	128.148	28	158.2364	27	91.3675	27
Q17	133.1326	47	161.4234	46	86.8378	46
Q18	149.7016	128	161.4578	126	90.3664	116
Q19	144.9722	104	161.9096	107	93.8922	97
Q20	154.4727	22	150.6469	22	91.5364	21
Q21	153.0729	247	193.3805	242	99.0206	258
Q22	159.1897	513	164.4872	476	105.1091	460

References

Belenguer, J.M., Martinez, M.C., Mota, E., 2000. A lower bound for the split delivery vehicle routing problem. Operations Research 48, 801–810.

Bertsekas, D.P., 1995. Dynamic programming and optimal control, vol. I. Athena Scientific, Belmont, Mass.

Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., 1995. Time constrained routing and scheduling. In: Network routingHandbooks Operation Research and Management Science, vol. 8. North-Holland, Amsterdam, pp. 35–139.

Dror, M., Trudeau, P., 1989. Saving by split delivery routing. Transportation Science 23, 141–145.

Dror, M., Trudeau, P., 1990. Split delivery routing. Naval Research Logistics 37, 383–402.

Dror, M., Laporte, G., Trudeau, P., 1994. Vehicle routing with split deliveries. Discrete Applied Mathematics 50, 239–254.

Frizzell, P., Griffin, J., 1995. The split delivery vehicle scheduling problem with time windows and grid network distances. Computers and Operations Research 22, 655–667.

Ho, S.C., Haugland, D., 2004. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. Computers and Operations Research 31, 1947–1964.

Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., Shmoys, D.B., 1985. The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization. John Wiley & Sons, New York.

Pearl, J., 1984. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, MA.

Sierksma, G., Tijssen, G.A., 1998. Routing helicopters for crew exchanges on off-shore locations. Annals of Operations Research 76, 261–286.