Pesquisa Operacional (2018) 38(2): 173-193

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
doi: 10.1590/0101-7438.2018.038.02.0173

A GENERALIZED DECOMPOSITION ALGORITHM FOR REAL-TIME TRUCK ROUTING PROBLEMS

Yihua Li[1], Qing Miao[2*] and Xiubin Bruce Wang[3]

Received April 29, 2017 / Accepted March 17, 2018

ABSTRACT. This paper is based on a practical project jointly conducted by a major trucking company and a renowned operations research consulting firm. It studies a large-scale, real-time truckload pickup and delivery problem. A number of cost factors are carefully measured such as loaded/empty travel distance, travel time, crew labor, equipment rental or operational cost, and revenue for completing the movements. This paper proposes a generalized decomposition algorithm that is capable of considering sophisticated business rules. The goal is to recommend executable and efficient truck routing decisions to minimize operating costs. Numerical tests are conducted with operational data from J.B.HUNT. A fleet of 5,000 trucks is considered in this experiment. The test result not only shows significant cost savings but also demonstrates computational efficiency for real-time application.

Keywords: Truck routing, decomposition algorithm, column generation.

1 INTRODUCTION

Research on vehicle routing has seen wide applications in the transportation industry such as truck, railway, airline and pipeline, which saves on costs while covers more demands. The objective is to improve the fleet operational efficiency. This research is important within the context of increasingly automated (or computerized) systems to support decision making for the routing/scheduling routines. It can be easily extended to solving other problems that may look seemingly different, but belong to the same family of NP-completeness (Li et al., 2010, 2012 and 2014). The social economic impact of this problem cannot be overestimated. In the past century, especially during the last 60 years, numerous research efforts have been made specially to solve the trucking industry's routing problems.

*Corresponding author.
[1]United Airlines, Willis Tower, 233 South. Wacker Drive, Chicago, IL 60606, United States.
E-mail: yihua.li@united.com
[2]The Home Depot, 2455 Paces Ferry Rd SE, Atlanta, GA 30339-1834, United States.
E-mail: qing_miao@homedepot.com
[3]Texas A&M University, College Station, TX 77843, United States. E-mail: bwang@tamu.edu

The main contribution of this paper is to introduce a decomposition algorithm based on a practical project. The goal is to recommend executable and efficient truck routing decisions to minimize operating costs. Numerical tests are conducted with operational data from J.B.HUNT. The test result not only shows significant cost savings but also demonstrates computational efficiency for real-time application.

Within the company, a fleet of vehicles are to be scheduled and routed to serve a known set of loads, and each load is considered as a full truckload with an origin and a destination (OD) associated with time window constraints for pickup and delivery. There are a finite number of drivers (or, crew in general) available to be assigned. Each driver has to return home within a period of 14 days according to the company rule as well as union regulation to retain drivers (for example, maximum consecutive hours of driving). A route consists of a sequence of moves to be fulfilled by a single vehicle. A typical route needs to have a starting location the same as termination location, which makes a tour to the associated vehicle. Depending on the service type, certain vehicles or drivers may not be eligible. For example, *Dedicated Contract Service* is a service primarily utilizes semi-trailer trucks to transport cargo across the country; *Intermodal Service* partners with railways, commercial airlines or port authorities to move containers. Furthermore, once the drivers are committed to certain loads, diversion is typically not allowed with the exception from management approval. For the convenient purpose, this study does not consider diversion in the problem formulation and presentation.

The primary objective is to minimize the operation cost. In addition to cost of labor, other major costs include fuel rate, travel distance, equipment rental and length of operation. Revenue from serving each load can be considered as a positive profit or negative cost whenever a load is covered or service is completed. Empty truck moves and empty driver moves (also called *deadhead* moves) do not generate revenue, thus are only considered costs to the company. It is expected that the developed algorithm is built into a decision making system to recommend routings automatically.

2 LITERATURE REVIEW

In contrast to the less-than-truckload where each vehicle carries multiple customer demands, truckload only allows a truck to serve a single customer demand each time. An example route of a vehicle starts from depot (or source), loads goods at a factory or warehouse, moves to the load destination facility, unloads the load before goes to another loading location and repeats the same truckload movement. In the end, the vehicle returns to the depot (Labadie & Prins, 2012). This is a typical vehicle routing/scheduling problem in literature.

There is a tremendous literature dealing with mathematical modeling of vehicle routing problems (VRPs), all of which imply that it is impossible to enumerate exhaustively all the possible routes for the vehicles. Earlier efforts to solving VRPs are summarized in extensive reviews as Desrochers et al. (1990) and Solomon (1987). Bramel et al. (1992, 1994) presented some probabilistic analyses of earlier heuristics for the deterministic version of the problem. Later, VRPs

with stochastic features drew more and more attention from the research community. These features included but were not limited to stochastic load distributions (Golden & Stewart, 1978; Stewart & Golden, 1983; and Bastian & Rinnooy Kan, 1992), stochastic travel time (Cook & Russell, 1978; and Berman & Simchi-Levi, 1989), or stochastic locations (Laporte et al., 1994; and Bertsimas & Howell, 1993). As the information technologies came into play, recent real-time and dynamic VRP problems became increasingly important (Yang et al., 2004). Powell et al. (1995) presented a survey of dynamic fleet optimizations dealing with some general issues. Later work of Powell et al. (2000) developed a practical model to consider dynamic assignment of drivers to known demands, which provided significant insights to our study problem here. Re-optimization policies are further introduced and tested in Yang, et al. (1998). The most recent reviews summarizing the state-of-art techniques are available in Laporte et al. (2013), Derigs et al. (2013), and Braekers et al. (2016).

Published articles on implementation, however, are less popular compared with the counterpart on theoretical studies. When it comes to practical VRP projects, people often look for details about how algorithms are developed and implemented. This paper is originated from industry projects within a major trucking company and focuses on proven practical techniques. Implementation details are revealed so that the readers can have a better understanding of the business logic. It aims at utilizing a combined optimization method and advanced information technology to develop a real-time dispatching system. The computational time invested in searching for better decisions in terms of shorter routes and more revenue should be cautiously balanced with the needs of coming up with a decision in a timely manner.

3 SOLUTION APPROACH

This decision support system requires a list of components to function properly. Information is updated via continuous data feed, stored in a centralized data warehouse. Forecast module provides projections on the existing truck moves and the estimation on the future demand. Prepossessing module turns the raw data into the format that are favored by the optimizer. Optimization component is the core module that handles the mathematical formulation and sophisticated algorithm.

3.1 Information Updating

Although the information arrives all the time, it is not required to run optimizer over the entire fleet all the time. Actually, there are two types of optimization jobs. First type of job is for planning purpose and runs daily. This is normally offline and is scheduled to run overnight or at meal breaks, so that the results are immediately available to dispatchers when they start work or resume work. Second type of job is for real-time optimization. This type of job only considers a smaller set of the fleet because most vehicles are already committed to their loads/route and do not need re-optimization. This is normally triggered from managing team whenever there is a need.

3.2 Forecast Engine

Certain loads are visible before they become available for pickup. For example, the containers can be moved by trains with a projected arrival time at the pickup location. The forecast engine would gather the information from partnered carriers to make reasonable projections on the availability of their own loads. This helps the optimization engine to look beyond the current demands and make informed decision for the immediate future. Beyond the visible horizon, long term forecast is also necessary to plan ahead in terms of fleet sizing and infrastructure change at strategic level.

3.3 Preprocessing

The preprocessor needs to selectively package the raw data into the network format that the downstream optimizer requires. The basic elements in this network are link (edge) and node (vertex), which are explained in section 5. Examining the feasibility of the links can reduce the burden on the optimizer. For example, it needs to filter out an infeasible link that tries to merry a wrong type of truck to a load. Each node represents an activity such as getting a truck, load or unload. After excluding infeasible links, certain business rules will further reduce the number of links by checking the distance, time or any other resource consumed between two nodes. If certain link violates resource limit, then this link is also excluded. The ultimate goal here is to allow the optimization module find reliable routes through the network easily.

3.4 Optimization Strategy

Due to the complicated resource constraints and business rules, it is inconvenient to formulate this truck routing problem into a network flow model. The alternative approach is to apply partition or set-covering model, where the objective function minimizes the combined cost of routes being selected.

An obvious difficulty here is the attempt to explicitly enumerate all the feasible routes, which are typically required in a partition model. Given the total number of loads and available drivers/trucks, the number of combination is extremely large and increases exponentially with the problem size. Additionally, the business rules are not easy to formulate, such as the maximum hours each driver can take in a tour. Each tour must start and end at driver's home terminal, and there are specified time windows for pickup and delivery. Here we propose a decomposition method to bypass the necessity of evaluating all possible combinations. During the iteration process, a master problem picks the best set of routes to minimize the total cost and *price* each load in terms of dual variables for the subproblem. The subproblem then updates the network and generates better routes for master problem to consider with. A schema that describes the entire framework of this column generation procedure is presented below in Figure 1. Here in this problem, one can assume that a column refers to a route.

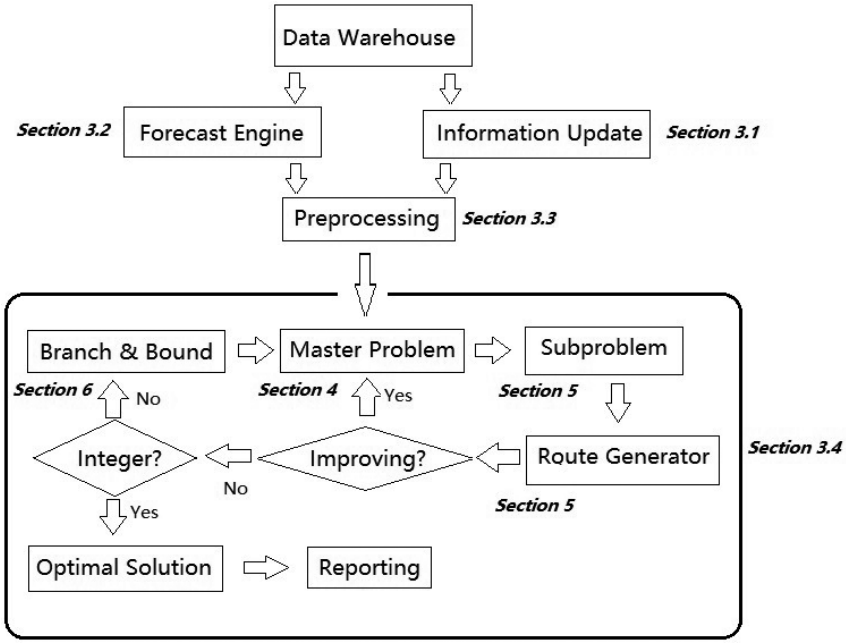

Figure 1 – Dependency Diagram and Flow Chart.

4 MASTER PROBLEM FORMULATION

$$\text{Minimize} \quad \sum_{j \in \Omega_k} c_j x_j \tag{1}$$

subject to:

$$\sum_{j \in \Omega_k} \delta_j^i x_j = 1 \quad \forall i \in N, \text{ dual } \pi_i \tag{2}$$

$$x_j = 0, 1 \quad \forall j \in \Omega_k \tag{3}$$

where:

- x_j decision variable, 1 if route j is selected, 0 otherwise
- Ω_k the set of routes in the k-th iteration of the column generation procedure
- c_j the cost associated with route j (operating cost – load revenue)
- N the set of loads
- δ_i^j if route j covers load i, 0 otherwise
- π_i dual associated i-th constraint (for load i)

The objective is to minimize the total cost to cover a known set of load demands. The revenue generated from moving a load is considered a negative cost when building the routes, and it usually causes the total route cost to be negative, otherwise the route would not be profitable. Equation (2) requires all the load movement to be assigned exactly once. Each load has a dual value π_i associated with it, which is the shadow price the load given the current master problem.

It is important to note that during the iteration, the master problem is solved as linear programming relaxation to get dual values. The side constraints are conditional and are not formulated into the master. For example:

$$\sum_{j \in \Omega_k} x_j \leq DR$$

Although the company prefers its own salaried drivers over the contracted drivers due to the lowered operating cost. DR is the total number of both. When the total number of routes being selected exceeds limit DR, it is intuitive to sort the route costs in an ascending order to pick the first DR. The uncovered loads can be either rejected or outsourced. This is to facilitate the construction of the subproblem, where all the dual values associated with Eq. (2) are utilized to reflect what master problem desires.

5 CONSTRAINED SHORTEST PATH SUBPROBLEM

Our subproblem tries to find an optimal path go through network G that does not consume more than limited resources, such as time window, duration of path, distance, etc. Because of the additional constraints applied to the path, the subproblem is therefore a *Constrained* Shortest Path Problem (CSPP). A path here represents a truck route in reality.

5.1 Subproblem Formulation

$$\text{Minimize} \sum_{(i,j) \in A} (c_{ij} - \pi_j) x_{ij}$$

where

- x_{ij} are the decision variables, 1 if node i is followed by node j, 0 otherwise
- c_{ij} the cost to proceed from node i to node j
- A the set of eligible connections, link $(i, j) \in A$
- π_j dual associated j-th constraint (for load j)

The construction of the subproblem is essential to the usefulness of the generated routes and the overall solution time. A route is a collection of links that are connected by nodes, it is also referred as a path through network. Since it has a single objective function to generate the most profitable route, different cost elements need to be normalized within the network. For example, the overall cost on link i to j is based on the load/empty factor, distance (i, j), the revenue of moving load j, and dual π_j for load j. Given the same distance, an empty-truck move has a baseline cost and zero revenue. A loaded-truck move may double the baseline cost but gain revenue. The changing value of π_j is passed from master problem to adapt to the current need.

Each route begins from the source S and ends at sink T. In most cases, the source and sink are the same physical location but have different time windows. This is due to the business rule that the driver has to return to home terminal when the route is completed. Starting from source S, each route has to connect to an artificial node where a driver is "picked up". Assigning a reasonable setup cost for such a node would encourage the model to minimize the total number of routes/drivers used.

Other resource constraints include (1) the maximum duration of each route, which is limited to 14 days; (2) the maximum combined travel distance for each route; (3) the time window for the load, which represents the earliest and latest time to pick up or deliver. In order to solve this multiple resource constrained shortest path problem with time windows, a specialized Label Setting Algorithm is applied.

5.2 The Algorithm for Shortest Path with Resource Constraints

Let (D_{ik}^*, C_{ik}) and (D_{jk}^*, C_{jk}) be the labels representing two different paths to node k. Then the first label **dominates** the latter, if and only if $(D_{ik}^*, C_{ik}) - (D_{jk}^*, C_{jk}) \geq (0, 0)$. The first label is **smaller than** the latter, i.e., $(D_{ik}^*, C_{ik}) \underset{<}{L} (D_{jk}^*, C_{jk})$, if and only if $(D_{ik}^*, C_{ik}) \neq (D_{jk}^*, C_{jk})$, and the first non-null element of $((D_{ik}^*, C_{ik}) - (D_{jk}^*, C_{jk}))$is positive. A label (D_k^*, C_k) is efficient if none of the labels at node k can dominate it. The path corresponding to an efficient label is defined as an efficient path. Only efficient labels and paths are kept.

Let Q_k be the set of labels associated with the cost lower bound of path ending at node $k \in V$, and P_k be the set of labels associated with feasible paths. The P_k defines the primal function. The primal function provides an upper bound on the cost of efficient solutions at node k. When primal and dual functions have same value for a given stage, the labels in P and Q are associated to an efficient path for the current stage.

Then the algorithm used to solve the subproblem is presented as follows:

Step 1. (Initialization).

$P_S = (0, 0, \ldots, 0)$, $Q_S = (0, 0, \ldots, 0)$, $P_i = \varnothing$ (empty), $Q_i = (a_i^1, a_i^2, \ldots, a_i^L, -\infty)$, $i \in V - S$

Find $(d_j^*, c_j) = \min_{(i,j)\in A} lex \left\{(d_{ij}^*, c_{ij})\right\}$, for each node $j \in V - S$. $O = \bigcup_{j\in V-S} (Q_j)$.

Step 2. (Calculation of the lexicographically smallest label in O).

If $O = \varnothing$**, stop**; [The current label(s) at the sink node T is (are) the shortest path(s)]. Otherwise, find the lexicographically smallest label $F(O) = \min_{(D_j^*, C_j)\in O} lex \left\{(D_j^*, C_j)\right\} = (D_j^{*'}, C_j)$.

Step 3. (Look for a label $(D_k^{*'}, C_k) \in B(O)$ to be treated).

$$B(O) = \left(D_k^*, C_k\right) \in O | F(O) \underset{\leq}{L} \left(D_k^*, C_k\right) \underset{<}{L} F(O) + \left(d_k^*, C_k\right).$$

Step 4. (Uncertainty zone).

Define the uncertainty zone for node k:

$$TR_k = \min lex\ b_k^*,\ D_k^* | \left(D_k^*, C_k\right) \in P_k \text{ and } D_k^* \underset{\geq}{L} D_k^{*'}.$$

Step 5. (Replacement of the label $\left(D_j^{*'}, C_j\right)$.

(A) Calculation of the efficient labels defining the dual function:

$$R_k = EFF\left\{\begin{array}{l}(D_k^*, C_k) = \left[(\max(a_k^l, D_i^l + d_{ik}^l), l = 1, 2, \ldots, L), C_i + C_{ik}\right] \\ |\forall(i,k) \in \mathrm{A}, (D_i^*, C_i) \in Q_i, \text{ and } D_k^* \text{ belonging to uncertainty zone}\end{array}\right\}$$

(B) Calculation of the feasible paths defining the primal function:

$$RP_k = EFF\left\{\begin{array}{l}(D_k^*, C_k) = \left[(\max(a_k^l, D_i^l + d_{ik}^l), l = 1, 2, \ldots, L), C_i + C_{ik}\right] \\ |\forall(i,k) \in \mathrm{A}, (D_i^*, C_i) \in P_i, \text{ and } D_k^* \text{ belonging to uncertainty zone}\end{array}\right\}$$

(C) Update sets: $P_k, Q_k : P_k \leftarrow P_k \cup (R_k \cap RP_k)$, $Q_k \leftarrow Q_k - (D_j^{*'}, C_j') \cup R_k$.

(D) $O \leftarrow O - (D_j^{*'}, C_j') \cup [R_k - (R_k \cap RP_k)]$.

(E) $B(O) \leftarrow B(O) - (D_j^{*'}, C_j')$. If $B(O) = \varnothing$, return to **Step 2**; otherwise return to **Step 3**.

A simple graph with two resource constraints (time, distance) and cost is presented in Figure 2 and Table 1, followed by an example to show how the algorithm works. Figure 2 shows the node and link connections in the network. Table 1(a) shows the drivers' profile. Table 1(b) shows loads' profile. Table 1(c) shows the constraint for the drivers and loads. In this example, the constraints are time window and mileage. Table 1(d) shows the resources being consumed on each link, as well as the cost on each link. Note that the actual route, which has multiple links, can be 14-days long and undertake more than just two or three loads.

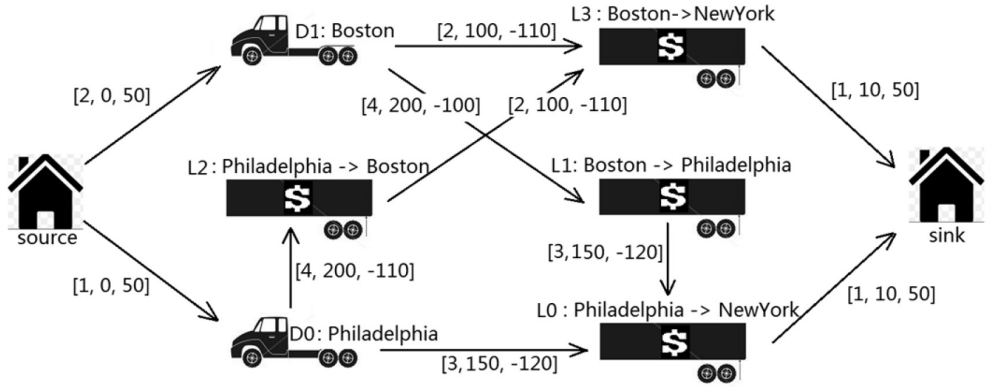

Figure 2 – Subproblem Network.

Where a_0 and b_0 are the beginning and ending pick-up times, and a_1 and b_1 are the minimum and maximum working miles, respectively.

Tables 1a-1d – Sample Database Tables.

(a)

Driver Information			
ID	Loc	Avail	Home
D0	Phila	07:00	NewYork
D1	Boston	07:00	NewYork
st. work hours <= 10, miles <= 500			
Ending at home			

(b)

Load Information			
ID	Pkup	Due	Dest
L0	Phila	1800	NewYork
L1	Boston	1200	Phila
L2	Phila	1200	Houston
L3	Boston	1800	NewYork
st. Delivery time <= Due			

(c)

Resource Constraints				
	a0	b0	a1	b1
S	0	0	0	0
D0	7	8	0	500
D1	7	8	0	500
L0	8	18	0	500
L1	7	12	0	500
L2	7	12	0	500
L3	8	18	0	500
T	12	19	0	500
	time window		min and max miles	

(d)

Resource Consumption			
Link	hours	miles	Cost-Rev
(S, D0)	2	0	50
(S, D1)	1	0	50
(D0, L0)	2	100	-110
(D0, L2)	4	200	-100
(D1, L1)	4	200	-110
(D1, L3)	3	150	-120
(L1, L0)	2	100	-110
(L2, L3)	3	150	-120
(L0, T)	1	10	50
(L3, T)	1	10	50

Deadhead is typically discouraged because moving an empty truck comes with a cost but there is no (direct) gain in revenue. Simply speaking, driver D_0 is originally located at Philadelphia and is eligible for Load L_0 and L_2 at the beginning. Once the load is delivered, the driver may become available again at the load destination (New York or Boston, depending on the load). Preprocessor skips the ineligible connections and adds eligible connections in term of links to the network. This preprocessing is a necessary step to reduce the problem size of the subproblem.

Step 1. Initialization.

$P_S = Q_S = (0, 0, 0)$, $P_{D0} = P_{D1} = P_{L0} = P_{L1} = P_{L2} = P_{L3} = P_T = \varnothing$,

$Q_{D0} = (7, 0, -\infty)$, $Q_{D1} = (7, 0, -\infty)$, $Q_{L0} = (8, 0, -\infty)$, $Q_{L1} = (7, 0, -\infty)$, $Q_{L2} = (7, 0, -\infty)$, $Q_{L3} = (8, 0, -\infty)$, $Q_T = (12, 0, -\infty)$,

$d_{D0} = (2, 0, 50)$, $d_{D1} = (1, 0, 50)$, $d_{L0} = (2, 100, -110)$, $d_{L1} = (4, 200, -110)$, $d_{L2} = (4, 200, -100)$, $d_{L3} = (3, 150, -120)$, $d_T = (1, 10, 50)$, and

$O = \{(7, 0, -\infty)_{D0}, (7, 0, -\infty)_{D1}, (7, 0, -\infty)_{L1}, (7, 0, -\infty)_{L2}, (8, 0, -\infty)_{L0}, (8, 0, -\infty)_{L3}, (12, 0, -\infty)_T\}$.

Step 2. $F(O) = (7, 0, -\infty)_{D0}$.

Step 3. $B(O) = \left\{(D_k^*, C_k) \in O | (7, 0, -\infty) \underset{\leq}{L} (D_k^*, C_k) \underset{<}{L} (9, 0, -\infty)\right\}$. We treat label $(7, 0 - \infty)$.

Step 4. Uncertainty zone $TR_{D0} = (8, 500)$.

Step 5. Replacement of $(7, 0, -\infty)$. (A) $R_{D0} = RP_{D0} = (7, 0, 50)$; (C) $P_{D0} = Q_{D0} = (7, 0, 50)$. $O = \{(7, 0, -\infty)_{D1}, (7, 0, -\infty)_{L1}, (8, 0, -\infty)_{L2}, (8, 0, -\infty)_{L0}, (8, 0, -\infty)_{L3}, (12, 0, -\infty)_T\}$. We treat label $(7, 0 - \infty)$.

Step 4. Uncertainty zone $TR_{D1} = (8, 500)$.

Step 5. Replacement of $(7, 0, -\infty)$. (A) $R_{D1} = RP_{D1} = (7, 0, 50)$; (C) $P_{D1} = Q_{D1} = (7, 0, 50)$. $O = \{(7, 0, -\infty)_{L1}, (7, 0, -\infty)_{L2}, (8, 0, -\infty)_{L0}, (8, 0, -\infty)_{L3}, (12, 0, -\infty)_T\}$, We treat label $(7, 0 - \infty)$.

Step 4. Uncertainty zone $TR_{L1} = (12, 500)$.

Step 5. Replacement of $(7, 0, -\infty)$. (A) $R_{L1} = RP_{L1} = (11, 200, -60)$; (C) $P_{L1} = Q_{L1} = (11, 200, -60)$. $O = \{(7, 0, -\infty)_{L2}, (8, 0, -\infty)_{L0}, (8, 0, -\infty)_{L3}, (12, 0, -\infty)_T\}$, We treat label $(7, 0 - \infty)$.

Step 4. Uncertainty zone $TR_{L2} = (12, 500)$.

Step 5. Replacement of $(7, 0, -\infty)$. (A) $R_{L2} = RP_{L2} = (11, 200, -50)$; (C) $P_{L2} = Q_{L2} = (11, 200, -50)$. $O = \{(8, 0, -\infty)_{L0}, (8, 0, -\infty)_{L3}, (12, 0, -\infty)_T\}$, We treat label $(8, 0 - \infty)$.

Step 4. Uncertainty zone $TR_{L0} = (18, 500)$.

Step 5. Replacement of $(8, 0, -\infty)$. (A) $R_{L0} = RP_{L0} = \{(9, 100, -60), (13, 300, -170)\}$; (C) $P_{L0} = Q_{L0} = \{(9, 100, -60), (13, 300, -170)\}$. $O = \{(8, 0, -\infty)_{L3}, (12, 0, -\infty)_T\}$, We treat label $(8, 0 - \infty)$.

Step 4. Uncertainty zone $TR_{L3} = (18, 500)$.

Step 5. Replacement of $(8, 0, -\infty)$. (A) $R_{L3} = RP_{L3} = \{(10, 150, -70), (14, 350, -170)\}$; (C) $P_{L3} = Q_{L3} = \{(10, 150, -70), (14, 350, -170)\}$. $O = \{(12, 0, -\infty)_T\}$, We treat label $(12, 0 - \infty)$.

Step 4. Uncertainty zone $TR_T = (18, 500)$.

Step 5. Replacement of $(12, 0, -\infty)$. (A) $R_T = RP_T = \{(10, 110, -10), (11, 160, -20), (14, 310, -120), (15, 360, -120)$; (C) $P_T = Q_T = \{(10, 110, -10), (11, 160, -20), (14, 310, -120), (15, 360, -120)$.

Step 2. $O = \varnothing$. **Stop**. Current solution $\{(10, 110, -10), (11, 160, -20), (14, 310, -120), (15, 360, -120)\}$. There are four shortest paths from the source node S to the sink node T respecting the resource constraints: (1) Path: $S \to D0 \to L0 \to T$ with cost -10; (2) Path: $S \to D1 \to L3 \to T$ with cost -20; (3) Path: $S \to D1 \to L1 \to L0 \to T$ with cost -120; (4) Path: $S \to D0 \to L2 \to L3 \to T$ with cost -120. Further examination would suggest that

Path 4 is dominated by Path 3. As a result, Paths 1, 2 and 3 are non-dominated optimal and are eligible to be added into the path/route pools in the master problem.

Note that in this example, labels in P and Q are always having the same sets of labels and paths because the resource constraints never get violated. In the case where certain paths exceed the resource limit, those parts are deemed as infeasible and therefore excluded at the current stage. The lower bound Q, however, would keep this infeasible label for future treatments. This technique is extremely useful when the resource on a directed link is negative. In other words, the previously infeasible path may become feasible again by adding consecutive links to it. This means that before all the paths are examined, one simply cannot determine the best or even most feasible paths.

6 THE SCHEME OF BRANCH AND BOUND (B&B)

Commercial software CPLEX is used to solve the master problem (linear programming relaxation). The optimal solution is generally non-integer (fractional). The B&B scheme is thus invoked and embedded into the column generation process to obtain an integer solution. The following are the details for the implementation (refer to the flow chart). If there is any existing feasible solution can be extracted from the current truck operating plan, then it should be utilized as the initial solution to speed up the search process.

6.1 The Branching Strategy

It is easy to observe that if the solution is non-integer, there must exist at least a pair of consecutive load nodes t_1, t_2 such that

$$0 < \sum_{j \in TR(t_1,t_2)} x_j < 1,$$

where $TR(t_1, t_2)$ is the set of routes in which t_2 is executed immediately after t_1. Based on the set $TR(t_1, t_2)$, the original problem is partitioned (branched) into two subproblems:

$$\text{0-branch}, \sum_{j \in TR(t_1,t_2)} x_j = 0, \quad \text{1-branch}, \sum_{j \in TR(t_1,t_2)} x_j = 1.$$

The testing shows that this strategy gives a more balanced search tree than default variable branching and generally finds an acceptable integer solution more quickly. Conceivably, this method is to branch on the relationship between two consecutive loads. It forces the link to be selected or to be eliminated in the subproblem.

6.2 How B&B Scheme is Embedded into the Column Generation Process?

When the B&B scheme is embedded into column generation process, a search tree is created. The nodes in the search tree (stored in a sorted queue) are treated one by one. If the queue is empty, the algorithm is terminated, and the optimal integer solution obtained so far is the solution we are seeking. By *treating* a node we mean that two branched nodes will be *created* from this node:

– one with $\sum_{j \in TR(t_1,t_2)} x_j = 0$ (left node),

– and the other with $\sum_{j \in TR(t_1,t_2)} x_j = 1$ (right node).

By *creating* a node we mean that the objective value and the solution at the node are found. Only the node corresponding to fractional solution with objective value smaller than that of the current optimal integer solution is inserted into the queue. Once a node corresponding to an integer solution is created and its objective value is smaller than that of the current optimal integer solution, then all untreated nodes in the queue with objective value not smaller than that of the created node will be pruned off. Whenever a node a created, a set of columns (for solving the master problem) and a network (for solving the subproblem) should be updated accordingly.

6.3 Information Storage and Retrieval

For solving the master problem and the subproblem at different nodes of the search tree, the information on columns and network must conform to the node to be created. There are two ways to get the information: one is from the root node; the other is from the node to be treated (parent node). With the first one, much more memory can be saved, but more computing time is needed (repeated computing), while with the second, the situation is reversed. There is a trade-off between the memory and the computing time. In the first case, we only need to store the original network and the columns generated at root node. The search tree is a binary tree and the root node is at 0 level. We use a label consisting of $(j + 1)$ identifiers to represent the location of a node at j-th level: $(j, j_1, j_2, \ldots, j_j)$, where the first one j is a digit (or digits) which represents the level number of the node location; the second one j_1 represents the location (left or right) of its ancestor at the first level; the third one j_2 represents the location of its ancestor at second level,..., and j_j represents its location at j-th level. $j_1, j_2, \ldots, j_j = L$ or R, L stands for left branch (0-branch); R stands for right branch (1-branch). For example, a node with label $(3, R, R, L)$ means that the node is at the third level of the search tree, and its ancestors are at the first and second levels on the right location. The third-level node is at left location. In the second case, we only need to indicate where the treated node is located and then modify the columns and network of its parent node respectively to process.

6.4 The Modification of the Columns for the Master Problem

In the 0-branch, we delete all columns i for which $a_{t_1,i} = 1$ and $a_{t_2,i} = 1$. In the 1-branch, we delete all columns i for which $a_{t_1,i} = 1$ and $a_{t_2,i} = 0$, or $a_{t_1,i} = 0$ and $a_{t_2,i} = 1$, and the row corresponding to constraint for node t_2 due to the redundancy. Figure 3 illustrates this process. The underlying assumption is that each load can be covered only once at most. The 2nd coverage for the same load will not bring additional revenue.

In 1-branch, the dimension of the basis matrix is thus reduced by one. The modification of the columns depends on the location of the created node in the search tree and also depends on from

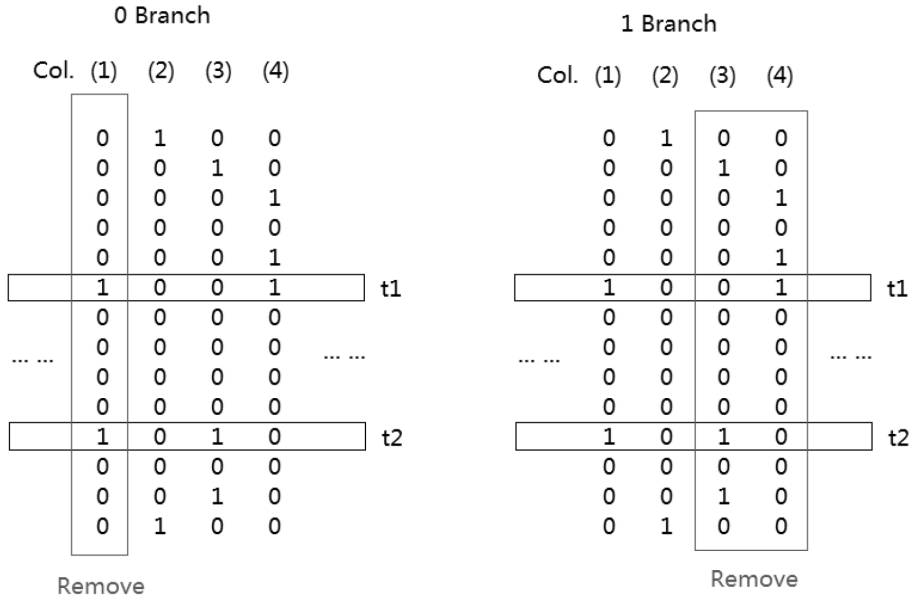

Figure 3 – Sample Branching.

where the information is obtained. For example, if we use the information on columns at root node and the created nodes are $(3, R, R, L)$, then all columns at root node having $a_{t_1,i} = 1$ and $a_{t_2,i} = 0$, $a_{t_1,i} = 0$ and $a_{t_2,i} = 1$, $a_{t_3,i} = 1$ and $a_{t_4,i} = 0$, $a_{t_3,i} = 0$ and $a_{t_4,i} = 1$, $a_{t_5,i} = 1$ and $a_{t_6,i} = 1$ are deleted. The row corresponding to constraint for node t_6 is also deleted due to the redundancy. We assume that the branching at first, second, and third level is based on $TR(t_1, t_2)$, $TR(t_3, t_4)$, $TR(t_5, t_6)$ respectively.

6.5 Modification of the Network for the Subproblem

We must restrict the columns to be generated by the subproblem to those that are compatible with the current created node in the search tree. The structure of the network used to generate the feasible columns should be modified accordingly. In the 0-branch, the columns covering consecutively nodes t_1 and t_2 are forbidden. As a reminder, there are the columns having t_2 executed immediately after t_1. In the network we split the middle node M, where the link corresponding to t_1 terminates and the link corresponding to t_2 starts, into two nodes M_1 and M_2; all links originally terminated at M are now moved to M_1, and all links originally started from M are moved to M_2. The resource constraints on M_1 and M_2 remain the same as M. In the 1-branch, we force any route covering t_1 to also cover t_2 immediately. In the network, the two links corresponding to t_1 and t_2 are condensed into one link, and all the links originally terminated at the middle node of the previous two links are deleted. Figure 4 shows this modification. The cost and resource consumption for the condensed link are: (a) the cost of the newly created link equals to cost(t_1)

+ cost(t_2), where cost(t_1) and cost(t_2) are the cost of t_1 and t_2, respectively; (b) the number of pieces of work still equals zero ; (c) the spread equals $t_1 + t_2$; (d) the work time equals $t_1 + t_2$.

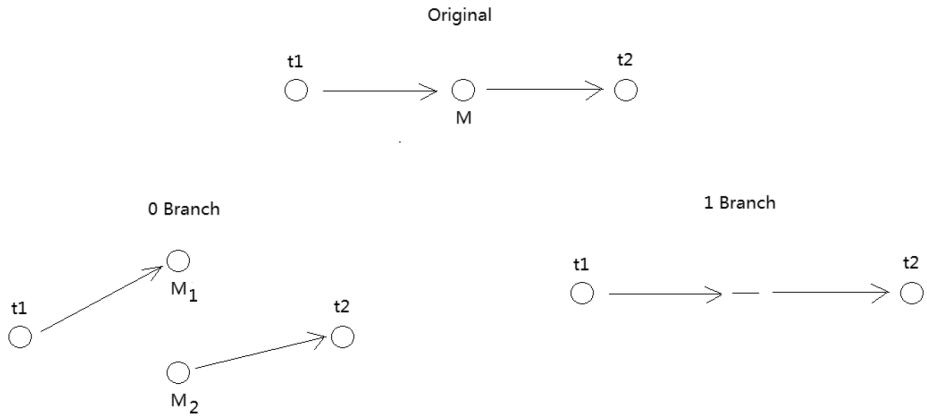

Figure 4 – Sample Network Modification.

With the same example, for the created node $(3, R, R, L)$ the modified network at this node is obtained from the original network. After splitting the two nodes (one between the links corresponding to t_1 and t_2, and the other between t_3 and t_4) into four nodes, and condensing two links corresponding to t_5 and t_6 into one, all the links originally terminated at the middle node of the links corresponding to t_5 and t_6 are deleted.

In the iteration process, a set of the dual values are passed from master problem. The cost (t_i) of the link i corresponding to load node t_i (i-th constraint) is subtracted by dual value π before the subproblem algorithm is executed. Once the desirable columns are generated, the cost of each link i is set to cost (t_i) (the original cost).

6.6 The Order of Choosing the Next Node and the Queue

The sum of fractional variables for a specified set $TR(t_1, t_2)$ and the objective values at each node should determine the next node to branch on. The node with smallest sum of fractions is chosen as a priority. If there is a tie, the one with the smaller objective value is chosen. If there is still a tie, then choose any arbitrary one. One should always keep a sorting queue (non-decreasing sequence of the sum of fractional variables) for the nodes to be treated and take the first one from the queue. When a fractional solution occurs, and its objective value is smaller than that of current optimal integer solution value, then the node corresponding to this fractional solution is inserted into the queue. Before inserting, we need to know the sum of fractions. From the fractional solution, one must find the first column $x_i < 1$ and a set $TR(t_1, t_2)$ consisting of two consecutive nodes t_1 and t_2 in column x_i such that

$$0 < \sum_{j \in TR(t_1, t_2)} x_j < 1;$$

otherwise, the process is repeated until such set is found.

7 TESTING RESULTS AND REMARKS

To prove the capability of the prototyped routing optimizer, we select top 10 US cities with the largest populations and set Dallas as home based depot (as shown in Table 2 and Figure 5).

Table 2 – Top 10 US Cities with the Largest Populations.

cityID	City	Population	Latitude	Longitude
1	New York	8,405,837	41	-74
2	Los Angeles	3,884,307	34	-118
3	Chicago	2,718,782	42	-88
4	Houston	2,195,914	30	-95
5	Philadelphia	1,553,165	40	-75
6	Phoenix	1,513,367	33	-112
7	San Antonio	1,409,019	29	-98
8	San Diego	1,355,896	33	-117
9	Dallas (depot)	1,257,676	33	-97
10	San Jose	998,537	37	-122

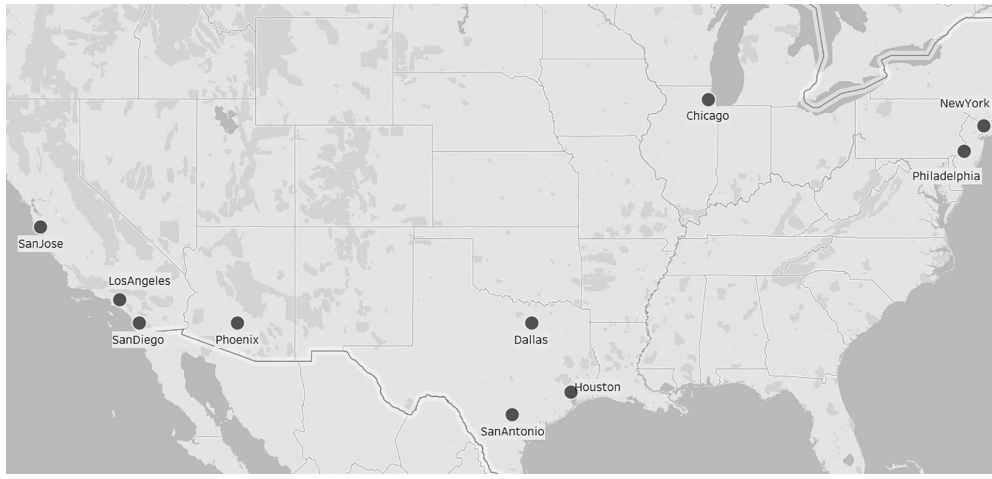

Figure 5 – Top 10 US Cities with the Largest Populations.

From these 10 cities we can have 45 non-directional city pairs sorted in an alphabet order on the origin city and destination city (as shown in Table 3).

For each pair, we call a random number between 0 and 1, if this number is less than 0.5, set a direction from origin to destination (ex. Houston to San Diego); otherwise set an opposite direction from destination to origin (ex. San Diego to Houston), where Dallas is a home base depot that every driver must return back to Dallas within 7000 miles (approximately equivalent to 2 weeks).

Table 3 – Forty-Five Non-directional City-Pairs.

odID	Orig	Dest	Miles	odID	Orig	Dest	Miles
1	Chicago	Dallas	967	24	Houston	San Jose	1885
2	Chicago	Houston	1083	25	Los Angeles	New York	2778
3	Chicago	Los Angeles	2016	26	Los Angeles	Philadelphia	2710
4	Chicago	New York	791	27	Los Angeles	Phoenix	373
5	Chicago	Philadelphia	758	28	Los Angeles	San Antonio	1353
6	Chicago	Phoenix	1735	29	Los Angeles	San Diego	120
7	Chicago	San Antonio	1242	30	Los Angeles	San Jose	341
8	Chicago	San Diego	2139	31	New York	Philadelphia	97
9	Chicago	San Jose	2163	32	New York	Phoenix	2409
10	Dallas	Houston	967	33	New York	San Antonio	1821
11	Dallas	Los Angeles	1436	34	New York	San Diego	2799
12	Dallas	New York	1547	35	New York	San Jose	2944
13	Dallas	Philadelphia	1467	36	Philadelphia	Phoenix	2344
14	Dallas	Phoenix	1065	37	Philadelphia	San Antonio	1742
15	Dallas	San Antonio	274	38	Philadelphia	San Diego	2735
16	Dallas	San Diego	1358	39	Philadelphia	San Jose	2911
17	Dallas	San Jose	1687	40	Phoenix	San Antonio	981
18	Houston	Los Angeles	1548	41	Phoenix	San Diego	355
19	Houston	New York	1627	42	Phoenix	San Jose	711
20	Houston	Philadelphia	1548	43	San Antonio	San Diego	1276
21	Houston	Phoenix	1174	44	San Antonio	San Jose	1692
22	Houston	San Antonio	197	45	San Diego	San Jose	460
23	Houston	San Diego	1468				

One of the performance measurements is Load Factor (LF). For each truck driver route, LF is the total loaded miles divide by total miles (loaded miles + empty miles) traveled starting from and returning to the depot: Dallas.

100 random generated data sets have been tested between current existing algorithm and our new column-generation based algorithm to compare both number of drivers used and Load Factor (LF) to complete each set of 45 loads crossing those 10 cities. Table 4 gives the details on each test case. The average run time is very stable and is within a few minutes for the tested problem size. Table 5 shows the overall performance.

Our sample testing indicates that our new column-generation based solution method will reduce about 16% of drivers and improve the Load Factor (LF) by about 9%.

The company also conducted an on-site experiment with real production data. About 3,500 externally contracted drivers, who were mainly independent business operators, and 1,500 corporate salaried drivers were considered in the problem for dispatching. Again, corporate drivers normally cost less than external contracts. About 15,000 loads are required to be covered. The

Table 4 – Computational Results of 100 Tests.

testID	Existing Method		New Method		Driver	LF
	num_drivers	LF	num_drivers	LF	Reduced (%)	Improved (%)
1	18	63	15	70	20	11.1
2	16	69	14	73	14.3	5.8
3	15	70	14	73	7.1	4.3
4	17	66	15	71	13.3	7.6
5	17	68	14	74	21.4	8.8
6	19	62	16	66	18.8	6.5
7	18	63	16	68	12.5	7.9
8	18	62	15	69	20	11.3
9	16	69	14	75	14.3	8.7
10	18	62	16	68	12.5	9.7
11	19	58	17	66	11.8	13.8
12	17	68	14	76	21.4	11.8
13	17	65	15	72	13.3	10.8
14	17	65	16	67	6.3	3.1
15	17	67	15	72	13.3	7.5
16	16	69	13	79	23.1	14.5
17	18	61	14	73	28.6	19.7
18	17	66	15	71	13.3	7.6
19	18	62	16	65	12.5	4.8
20	19	59	17	63	11.8	6.8
21	17	67	15	69	13.3	3
22	17	66	15	72	13.3	9.1
23	17	65	14	75	21.4	15.4
24	17	65	15	69	13.3	6.2
25	16	71	14	74	14.3	4.2
26	19	58	16	66	18.8	13.8
27	17	69	14	77	21.4	11.6
28	17	66	15	74	13.3	12.1
29	17	65	15	73	13.3	12.3
30	16	69	15	71	6.7	2.9
31	18	63	14	75	28.6	19
32	17	65	14	74	21.4	13.8
33	18	61	15	68	20	11.5
34	18	61	15	69	20	13.1
35	16	72	14	73	14.3	1.4
36	18	64	16	68	12.5	6.3
37	18	61	16	66	12.5	8.2
38	18	64	16	67	12.5	4.7
39	17	65	15	71	13.3	9.2
40	17	63	16	67	6.3	6.3
41	16	68	14	76	14.3	11.8
42	17	66	15	69	13.3	4.5

Table 4 (continuation).

testID	Existing Method		New Method		Driver	LF
	num_drivers	LF	num_drivers	LF	Reduced (%)	Improved (%)
43	17	68	14	77	21.4	13.2
44	17	68	14	72	21.4	5.9
45	16	69	13	76	23.1	10.1
46	16	71	14	74	14.3	4.2
47	18	62	15	71	20	14.5
48	16	69	15	69	6.7	0
49	17	65	13	77	30.8	18.5
50	17	66	14	72	21.4	9.1
51	18	64	16	68	12.5	6.3
52	17	68	14	73	21.4	7.4
53	18	59	17	62	5.9	5.1
54	17	63	16	64	6.3	1.6
55	19	60	17	63	11.8	5
56	17	65	14	72	21.4	10.8
57	17	67	14	74	21.4	10.4
58	19	62	16	66	18.8	6.5
59	18	64	16	67	12.5	4.7
60	18	63	15	68	20	7.9
61	17	64	15	70	13.3	9.4
62	18	62	15	69	20	11.3
63	17	65	15	70	13.3	7.7
64	18	61	14	73	28.6	19.7
65	17	64	15	71	13.3	10.9
66	19	58	18	59	5.6	1.7
67	17	68	14	76	21.4	11.8
68	18	64	14	73	28.6	14.1
69	16	71	13	78	23.1	9.9
70	17	64	13	79	30.8	23.4
71	19	61	17	64	11.8	4.9
72	17	68	14	75	21.4	10.3
73	18	63	15	70	20	11.1
74	18	60	18	60	0	0
75	17	65	14	74	21.4	13.8
76	17	68	15	72	13.3	5.9
77	17	63	16	65	6.3	3.2
78	18	65	15	71	20	9.2
79	17	69	13	80	30.8	15.9
80	17	66	13	77	30.8	16.7
81	16	70	15	71	6.7	1.4
82	18	66	15	71	20	7.6
83	18	62	14	72	28.6	16.1
84	17	66	16	68	6.3	3

Table 4 (continuation).

testID	Existing Method		New Method		Driver	LF
	num_drivers	LF	num_drivers	LF	Reduced (%)	Improved (%)
85	18	64	15	71	20	10.9
86	17	65	15	71	13.3	9.2
87	16	67	15	73	6.7	9
88	16	71	14	73	14.3	2.8
89	18	65	13	78	38.5	20
90	16	70	16	67	0	-4.3
91	17	63	14	74	21.4	17.5
92	17	68	15	70	13.3	2.9
93	17	64	15	71	13.3	10.9
94	17	66	14	76	21.4	15.2
95	17	65	14	73	21.4	12.3
96	19	60	17	64	11.8	6.7
97	17	66	14	72	21.4	9.1
98	19	60	17	64	11.8	6.7
99	18	62	15	70	20	12.9
100	17	67	14	73	21.4	9

Table 5 – Comparison Between Existing And New Methods.

Existing Method		New Method		Driver	LF
avg_drivers	avg_LF	avg_drivers	avg_LF	Reduced (%)	Improved (%)
17.3	65	14.9	70.9	16.3	9.1

decision variable is defined as a route, which is the combination of drivers, the loads and sequence of covering the loads. Without any of the preprocessing to eliminate routes, the number of decision variables is 2.5×10^{20} if the maximum of four loads are allowed in a single route. This number increases to 3.8×10^{24} if five loads are allowed in a single route. The prototype successfully considered all the major business criteria and constraints to have problem solved within 20 minutes on the company's mainframe machine. After comparing with the company's then-current operating plan, over 10% cost saving was achieved, which amounted to millions of dollars annually.

The proposed decomposition algorithm can be easily generalized to many other industries that share similar characteristics. For example, railway, airlines and pipeline share the similar network characteristics. Resource constraints can be modeled in terms of time, materials, vehicle or line capacity, distances, volumes or even headcount. Sometimes it is difficult to consider certain requirements such as recurring maintenance constraints and sophisticated union rules, solving the problem without these rules as in the case of this study at least can serve as a benchmark for the real problem. We leave as a future potential research effort to incorporate those additional constraints into routes developed.

REFERENCES

[1] BASTIAN C & RINNOOY KAN AHG. 1992. The Stochastic Vehicle Routing Problem Revisited. *European Journal of Operational Research*, **56**: 407–412.

[2] BERMAN O & SIMCHI-LEVI D. 1989. The Traveling Salesman Location Problem on Stochastic Networks. *Transportation Science*, **23**: 54–57.

[3] BERTSIMAS DJ & HOWELL LH. 1993. Further Results on the Probabilistic Traveling Salesman Problem. *European Journal of Operational Research*, **65**: 68–95.

[4] BRAEKERS K, RAMAEKERS K & VAN NIEUWENHUYSE I. 2016. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, **99**: 300–313.

[5] BRAMEL J, COFFMAN JR EG, SHOR P & SIMCHI-LEVI D. 1992. Probabilistic Analysis of Algorithms for the Capacitated Vehicle Routing Problem with Unsplit Demands. *Operations Research*, **40**: 1095–1106.

[6] BRAMEL J, LI CL & SIMCHI-LEVI D. 1994. Probabilistic Analysis of the Vehicle Routing Problem with Time Windows. *American Journal of Mathematical and Management Science*, **13**: 267–322.

[7] COOK TM & RUSSELL RA. 1978. A Simulation and Statistical Analysis of Stochastic Vehicle Routing with Timing Constraints. *Decision Science*, **9**: 673–687.

[8] DERIGS U, PULLMANN M & VOGEL U. 2013. Truck and Trailer Routing-Problems, Heuristics and Computational Experience. *Computers and Operations Research*, **40**: 536–546.

[9] DESROCHERS M, LENSTRA JK & SAVELSBERGH MWP. 1990. A Classification Scheme for Vehicle Routing and Scheduling Problems. *European Journal of Operational Research*, **46**: 322–332.

[10] GOLDEN BL & STEWART WR. 1978. Vehicle Routing with Probabilistic Demands. HOGBEN D & FIFE D. (eds.). *Computer Science and Statistics: Tenth Annual Symposium on the Interface*. NBSS Special Publication, National Book Service, Toronto, Canada, pp. 252–259.

[11] LABADIE N & PRINS C. 2012. Vehicle Routing Nowadays: Compact Review and Emerging Problems. In: *Production System and Supply Chain Management in Emerging Countries: Best Practices*, Selected Papers From the International Conference on Production Research (ICPR). Springer, Berlin, pp. 141–166.

[12] LAPORTE G, LOUVEAUX FV & MERCURE H. 1994. A Priori Optimization of the Probabilistic Traveling Salesman Problem. *Operations Research*, **42**: 543–549.

[13] LAPORTE G, TOTH P & VIGO D. 2013. Vehicle Routing: Historical Perspective and Recent Contributions. *European Journal of Operational Research*, **1**: 1–4.

[14] LI Y, MIAO Q & WANG X. 2014. High-Speed Train Network Routing with Column Generation. *Transportation Research Record: Journal of the Transportation Research Board*, **2466**: 58–67.

[15] LI Y, MIAO Q & WANG X. 2010. A Column Generation Method for the U.S. Army Logistics Air Fleet Scheduling. In: *Transportation Research Record: Journal of the Transportation Research Board*, **2197**: 36–42.

[16] LI Y, MIAO Q & WANG X. 2012. U.S. Postal Airmail Routing Optimization. In: *Transportation Research Record: Journal of the Transportation Research Board*, **2234**: 21–28.

[17] POWELL WB, JAILLET P & ODONI A. 1995. Stochastic and Dynamic Networks and Routing. BALL MO, MAGNANTI TL, MONMA CL & NEMHAUSER GL. (eds.). *Handbooks in Operations Research and Management Science*, **8**. Network Routing. Elsevier (North-Holland), Amsterdam, pp. 141–296.

[18] POWELL WB, SNOW W & CHEUNG RK. 2000. Adaptive Labeling Algorithms for the Dynamic Assignment Problem. *Transportation Science*, **34**: 50–66.

[19] SOLOMON MM. 1987. Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research*, **35**: 254–265.

[20] STEWART WR & GOLDEN BL. 1983. Stochastic Vehicle Routing: A Comprehensive Approach. *European Journal of Operational Research*, **14**: 371–385.

[21] YANG J, JAILLET P & MAHMASSANI H. 2004. Real-Time Multivehicle Truckload Pickup and Delivery Problems. *Transportation Science*, **38**(2): 135–148.

[22] YANG J, JAILLET P & MAHMASSANI HS. 1998. On-line Algorithms for Truck Fleet Assignment and Scheduling Under Real-time Information. *Transportation Research Record*, **1667**: 107–113.