TRUCKLOAD CONTINUOUS MOVE OPTIMIZATION

Manoj Lohatepanont, Sc.D.
Department of Civil Engineering
Chulalongkorn University
Bangkok 10330, THAILAND
E-Mail: LManoj@gmail.com

Yossiri Adulyasak
Department of Civil Engineering
Chulalongkorn University
Bangkok 10330, THAILAND
E-Mail: a.yossiri@gmail.com

Abstract The problem of excessive empty backhaul distances is a major challenge in the planning and operation of truckload transportation, in which goods are picked up from an origin and delivered to a destination without mid-route pickups or deliveries. Backhaul distances can be reduced by combining two or more truckload trips together to form a sequence of *continuous move* truckload trips. Finding optimal or effective combinations of truckload trips, however, is complicated, especially for large-scale transportation network, because the number of possible combinations increases exponentially with the number of shipments. In this paper, we propose a continuous move optimization model for large-scale transportation network, incorporating major operational complexities, namely, fleet-commodity compatibility, trip-based cost function, and time windows. We develop two solution approaches—an exact column-generation-based branch-and-bound algorithm and a heuristic algorithm—which yield significant empty haul distance reduction under relatively short runtimes, and provide a comparison study measuring the effectiveness and applicability of the two methods.

Keywords: Vehicle Routing, Lane Covering, Continuous Move, Large-Scale Optimization, Column Generation

1. PROBLEM DESCRIPTION

This paper focuses on Truckload (TL) operation, in which trucks pickup shipments from their origins and directly deliver them to their destinations. On their return trips, if backhaul loads cannot be secured, the trucks return empty. There are a few commonly used countermeasures, e.g., back-haul and continuous-move, each of which offers varying degree of effectiveness depending on the characteristics of the demand and the supply.

In the *back-haul* operation, the carrier attempts to secure a shipment from the current location back to the base. Back-haul operation is effective when there is a reasonable balance of the transportation demand going in both directions within some acceptable time windows and utilizing the same type of vehicles. Like the back-haul operation, the *continuous-move* operation attempts to reduce empty-haul distance by carrying shipments on the way back to the base. But unlike the back-haul operation, the continuous-move operation attempts to match shipments more extensively by looking beyond the exact opposite OD direction. Specifically, it allows a sequential continuous movement of the truck, visiting two or more destinations in the sequence before ultimately returning to the base. This improves upon the back-haul operation by allowing shipments to be matched not necessarily in the exact opposite OD direction and sequenced in a more flexible manner. That is, it does not rely heavily on the balance of opposite OD shipments like the back-haul operation does. This increased operational efficiency comes at a great planning complexity. Specifically, planners now have exponentially many ways they can match shipments to create efficient *continuous-move trips* (c-move trips).

In this paper, we study a continuous move planning optimization model for large-scale transportation network, incorporating major operational complexities such as fleet-commodity compatibility, trip-based cost function, and time windows.

2. MODEL

In this section, we state the formal problem statement, present model formulation, and outline our approach to modeling operational complexities.

2.1. Problem Statement

The *Continuous-Move Optimization (CMO) Problem* can be formally stated as follows:

> *Given a truckload distribution network comprising of multiple fleet types, origins, destinations, predetermined routes connecting the origins to/from the destinations, trip-based cost functions, operational time windows at origins and destinations, and truckload trip demands over the network, find the cost minimizing continuous move plan satisfying all truckload trips demanded.*

2.2. Notations

To facilitate our discussion, we summarize all notations used in the formulation and present them here.

Variables

x_{p_k} is the number of times a fleet-type-k c-move trip p_k is used, or, equivalently, the number of fleet-type-*k* trucks making the fleet-type-*k* c-move trip p_k.

n_k^i is the number of fleet-type-*k* trucks used at an origin *i*.

Parameters

c_{p_k} is the trip-based cost of a fleet-type-*k* c-move trip p_k.

r_k^i is the daily cost of using a fleet type *k* truck from an origin *i*.

$\delta_{p_k}^{ij}$ is the number of times a c-move trip p_k covers a fleet-type-*k* OD shipment t_k^{ij}.

$\alpha_{p_k}^i$ equals 1 if a fleet-type-*k* c-move trip p_k begins the day at an origin *i*, and 0 otherwise.

T_k^{ij} is the number of fleet-type-*k* shipments demanded from *i* to *j*.

N_k^i is the number of fleet-type-*k* trucks available at the beginning of the day at an origin *i*.

Sets

P_k is the set of fleet-type-*k* c-move trips, indexed by p_k.

I is the set of origins, indexed by *i*.

J is the set of destinations, indexed by *j*.

O is the set of origin-destination (O-D) pairs, indexed by *ij*.

2.3. Mathematical Formulation

The authors formulate this continuous move optimization model as a set-partitioning model where the set comprises OD shipments and each partition is an ordered sequence of OD

shipments in a c-move trip. The objective of the model is to find the cost-minimizing partitioning of the OD shipments.

Notice that the main advantage of modeling the CMO problem as a set partitioning model is that each variable, x_{p_k}, represents a continuous move trip covering shipments $ij \in O$, $\delta_{p_k}^{ij}$ times. Thus, we can model any complicated trip-based cost function very easily through the objective function coefficient, c_{p_k}. In addition, if any cost dominance exists among c-move trips visiting similar stops but in different order, we can easily remove dominated trips from consideration. The disadvantage of set partitioning model, however, is the exponential number of the variables/c-move trips. We, therefore, need to develop special algorithms to solve this problem efficiently. The solution algorithms will be presented later in Section 3.

The *Continuous Move Optimization (CMO) Model* is formulated as follows:

$$Min \quad \sum_{k \in K} \sum_{p_k \in P_k} c_{p_k} x_{p_k} + \sum_{k \in K} \sum_{i \in I} r_k^i n_k^i \tag{1}$$

Subject to:

$$\sum_{p_k \in P_k} \delta_{p_k}^{ij} x_{p_k} = T_k^{ij} \qquad \forall k \in K, \forall ij \in O \tag{2}$$

$$\sum_{p_k \in P_k} \alpha_{p_k}^i x_{p_k} - n_k^i = 0 \qquad \forall k \in K, \forall i \in I \tag{3}$$

$$x_{p_k} \geq 0 \quad \text{and integer} \qquad \forall k \in K, \forall p \in P_k \tag{4}$$

$$0 \leq n_k^i \leq N_k^i \quad \text{and integer} \qquad \forall k \in K, \forall i \in I\,. \tag{5}$$

The first term in the objective function (Equation 1) minimizes the summation of trip-based costs and the second term minimizes the number of vehicles used. Constraints (2) dictate that all shipments from origin i to destination j, $ij \in O$, by fleet type $k \in K$ are covered. Constraints (3) count the number of vehicles of fleet type $k \in K$ originating from origin $i \in I$, which cannot exceed N_k^i controlled by Constraints (5). Constraints (4) and (5) ensure non-negative integrality.

Recall from Section 2.2 that we *a priori* assign shipments/commodities to fleet types. Thus, the original problem decomposes into $|K|$ subproblems and the complexity of the problem is greatly reduced because we can solve the problem for each fleet type independently. For subproblem $k \in K$ the formulation reduces to:

$$Min \quad \sum_{k \in K} \sum_{p_k \in P_k} c_{p_k} x_{p_k} + \sum_{k \in K} \sum_{i \in I} r_k^i n_k^i \tag{6}$$

Subject to:

$$\sum_{p_k \in P_k} \delta_{p_k}^{ij} x_{p_k} = T_k^{ij} \qquad \forall ij \in O \tag{7}$$

$$\sum_{p_k \in P_k} \alpha_{p_k}^i x_{p_k} - n_k^i = 0 \qquad \forall i \in I \tag{8}$$

$$x_{p_k} \geq 0 \quad \text{and integer} \qquad \forall p \in P_k \tag{9}$$

$$0 \leq n_k^i \leq N_k^i \quad \text{and integer} \qquad \forall i \in I\,. \tag{10}$$

2.4. Modeling Operational Complexities

We take into consideration a number of operational constraints, namely, fleet-commodity compatibility, trip-based cost function, and time windows, in order to ensure the practicality of that the solution.

2.4.1 Fleet-Commodity Compatibility

The fleet-commodity compatibility is important because different shipments/commodities require different handling equipments and types of vehicle. To handle this requirement, we decompose our problem into subproblems by fleet type. That is, we *a priori* assign commodities/shipments to fleet types and optimize the continuous-move plan by solving the CMO problem for each fleet type independently. The fleet-commodity assignment can be done manually by planners or heuristically using a simple algorithm that observes (i) the fleet-commodity compatibility requirement and (ii) the approximate fleet size constraints.

2.4.2 Trip-Based Cost Function

The typical cost function of a TL operation trip is straightforward and often is driven primarily by distance and/or weight of the OD shipment. In a continuous-move operation, however, the cost function can be more complicated. The costs of the heavy-haul legs of the trip can be calculated accurately by measuring the distances and/or weights of the OD shipments, but the costs of the empty-haul legs of the trip cannot be fixed until the selection and the order of the heavy-haul legs of that c-move trip is determined. If the OD shipments making up the c-move trip or the order of the OD shipments change, the total cost of the trip can change. Thus, a trip-based cost function is necessary.

Because each variable in the CMO model represents an entire c-move trip, its objective function coefficient can represent profit or cost of that particular c-move trip. Any complex trip-based profit or cost function can be adopted.

In this paper, we minimize the total cost of the operation, and our cost function comprises four components: rental, distance-based, weight-based, and driver costs. Rental cost is the actual rent or the estimated depreciation of the truck in Baht per vehicle per day. Distance-based costs are measured in Baht per km and include such items as fuel, oil, and maintenance costs. Weight-based costs, in Baht per ton-km, are levied on some high density shipments. Driver costs are paid per shipment on a decreasing scale, i.e., on subsequent shipments (after the first shipments), drivers's pays are reduced by some small percentages.

2.4.3 Time Windows

We model three types of time windows in our model: (i) pickup, (ii) delivery, and (iii) truck-ban time windows. Pickup and delivery time windows represent the time windows at the origins and destinations, within which trucks can make pickups and deliveries. The model takes into consideration also the loading and unloading times at the origins and destinations. Truck ban time windows are specific times of day that large trucks are banned on the streets in the heart of Bangkok—a traffic management policy to reduce congestion in center of the city during peak hours.

In this section, we describe how we model the pickup and delivery time windows and explain the algorithm used to check the time-window feasibility of a c-move trip. We first describe the notations below.

k	is the max number of shipments allowed in a c-move trip.
e_i	is the window opening time at Location i (e_o is at the origin).
l_i	is the window closing time at Location i (l_o is at the origin).
m	is the number of truck-ban periods.
et_i	is the window opening time at Location i (e_o is at the origin).
lt_i	is the window closing time at Location i (l_o is at the origin).
d_o	is the departure time at the origin.
a	is the current arrival time.
t_i	is the total travel time from location i, including loading/unloading time (if any).
s	is the departure time shift from e_o, i.e., the difference between d_o and e_o.
s_i	is the arrival time shift at Location i.
f_s	is the trip flexibility (the available time window for waiting time reduction).
w_i	is the waiting time at Location i.
w	is the total waiting time.
TE	is the end-of-trip time back at the origin.

To understand, consider Figure 1. On the left side of the figure, the solid arrow represents a truck moving from Location 0 (origin) to Location 1 (destination). At Location 0, it has a time window of $l_o - e_o$ and at Location 1, the time window width is $l_1 - e_1$. The truck leaves Location 0 at time $d_o = e_o + s$ to arrive at Location 1 at time $a = d_o + t_o$, where t_o is the total travel time from Location 0, including loading/unloading time (if any). If the truck had left earlier than $e_o + s$, it would have to wait at Location 1. At Location 0, f_s measures the trip flexibility, i.e., the remaining time flexibility of the trip to move from the current departure time d_o at the truck origin (P1).

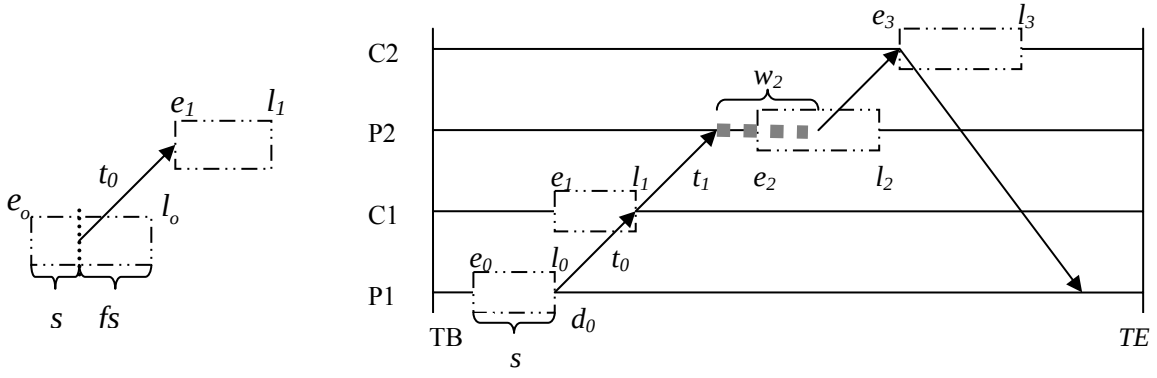

Figure 1 Graphical representation.

The right side of Figure 1 demonstrates an example of a trip leaving the origin (P1) at time $d_o = l_o$, arriving at the first customer (C1) at time l_1. The truck then departs C1 and arrives at P2 before the window opening time e_2 at P2. It, therefore, has to wait w_2 minutes. The truck

leaves from P2 to arrive at the window opening time e_3 at C2. Then, it returns to the origin before the ending time *TE*.

```
1: d ← e_o
2: f_s ← l_o − e_o
3: s ← 0
4: for i = 1 to 2k do
5:   d ← Adjust_Time_for_Truck_Ban
6:   s_i ← 0
7:   a ← d + t_{i−1}
8:   d ← a
9:   w ← 0
10:      if ( d > l_i ) then break        //Infeasible Trip
11:      if ( d ≤ e_i ) then
12:           s_i = min ( e_i − a,  f_s )
13:           w_i = max ( e_i − a −  f_s , 0)
14:           w = w + w_i
15:           f_s = min ( l_i − e_i , f_s − s_i )
16:           s = s + s_i
17:           d = e_i
18:      endif
19:   f_s = min ( l_i − d , f_s )
20:   d_o = e_o + s
21: endfor
```

Figure 2 Pseudo code for verifying the time window feasibility of a c-move trip.

Figure 2 gives the pseudo code of the algorithm we developed (adapted from the work of Ergun, Kuyzu, and Savelsbergh, 2005) to verify time window feasibility of a c-move trip. `Lines 1-3` initiate parameters d, f_s, and s. The algorithm then iteratively checks the arrival and departure time of each subsequent destination in the c-move trip considered (`Line 4`). `Line 5` calls on a sub-function (Figure 3) to adjust departure times to avoid truck-ban windows (Figure 4).

```
1: for j = 0 to m do
2:     if ((d + t_{i-1} < lt_i or d > lt_i) = FALSE) then
3:         d = lt_i
4:     endif
5: endfor
6: return d
```

Figure 3 Pseudo code for adjusting departure time to avoid truck bans.

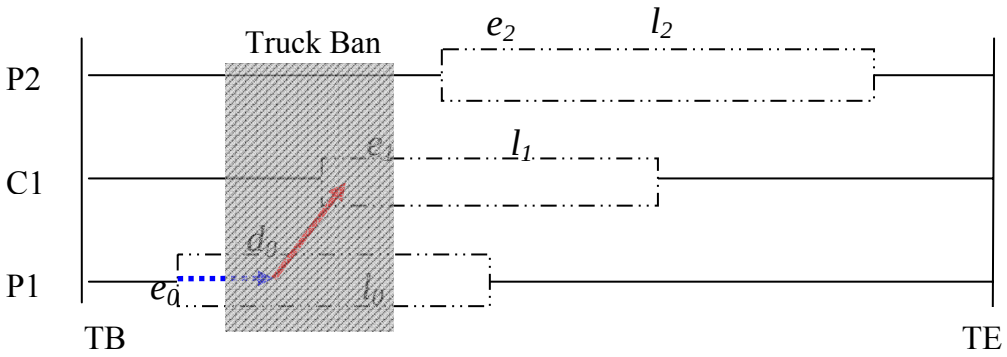

Figure 4 An example of a truck traveling inside a truck-ban period..

`Lines 6-9` initiate the temporary parameters for each iteration. `Line 10` identifies infeasibility at Location *i* (see Figure 5) when the earliest departure time ($d \leftarrow a$) is greater than the window closing time (l_i), in which case it marks the trip infeasible and removes it from further consideration.

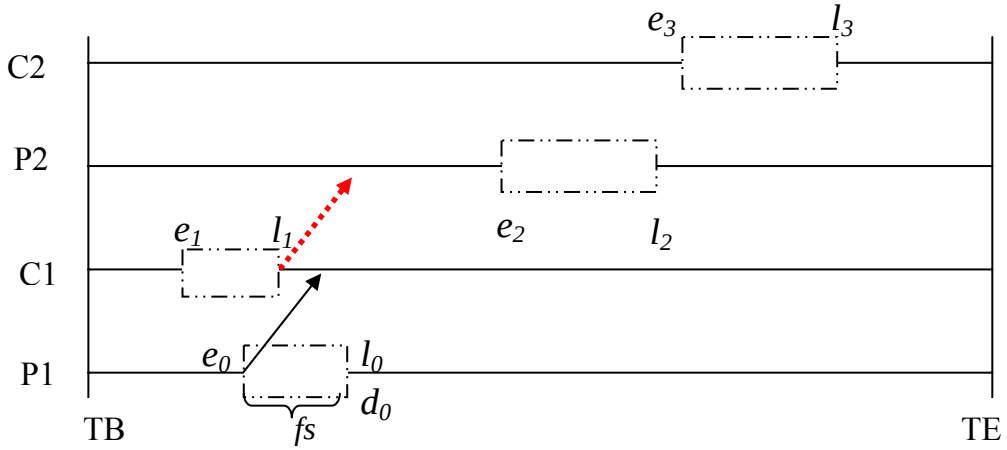

Figure 5 An example of time window infeasibility.

If the arrival time at Location *i* (C1) is earlier than the earliest time (e_i) (`Line 11`), the arrival time can be shifted by the smaller of $e_i - a$ (Figure 6) and f_s (Figure 7) (`Line 12`) in order to minimize the total waiting time accrued during the trip (*w*). The total trip waiting time is the summation of the waiting time at each location, which is the larger of $e_i - a - f_s$ and 0 (`Line 13`). `Line 14` updates the total waiting time.

The new trip flexibility changes to be the smallest of $l_i - e_i$, $l_i - d$, and $f_s - s_i$ as the truck moves along its trip (`Line 15`). And the new initial departure time (d_o) shifts by the amount *s*, which must be updated at each iteration by the addition of an amount equals to s_i (`Line 16`). The departure from Location *i* (d_i) takes places at the earliest time (`Line 17`).

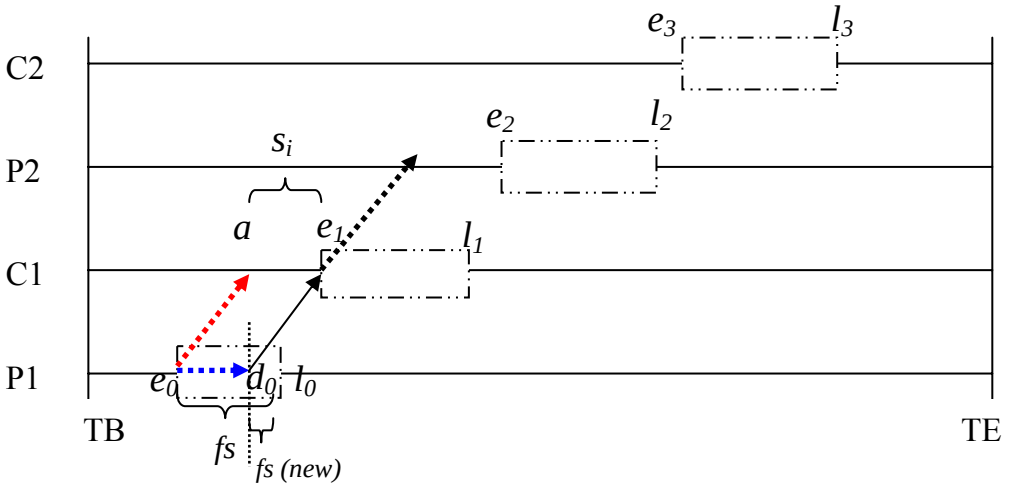

Figure 6 An early arrival is shifted to the earliest time at Location *i*.

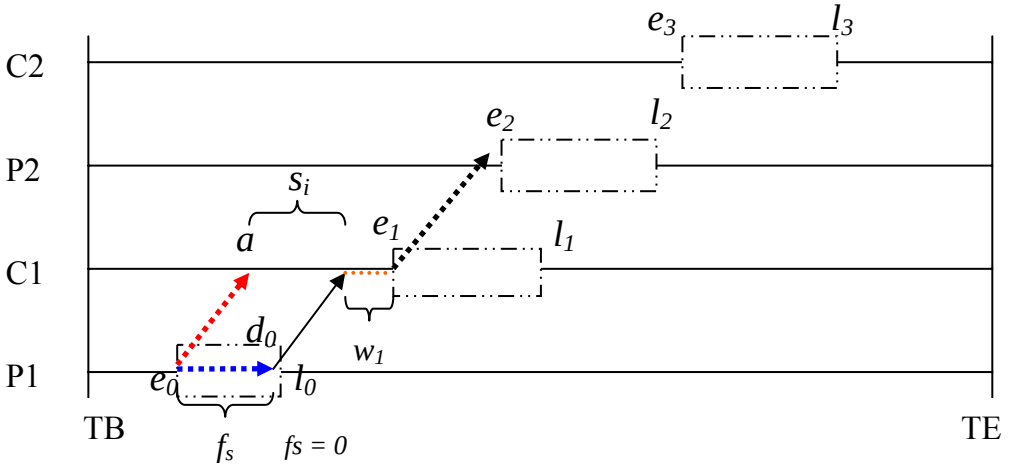

Figure 7 An early arrival is shifted by the remaining trip flexibility.

Figure 8 demonstrates the case when the arrival time at Location *i* is within the time window at Location *i*. In this case, the new trip flexibility is the smaller of $l_i - d$ and f_s (`Line 19`), and the new initial departure time need not change (`Line 20`).

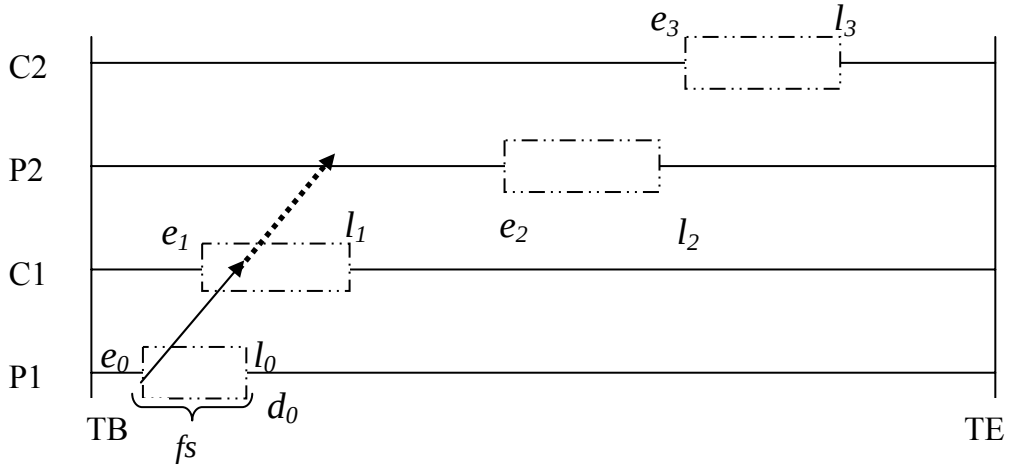

Figure 8 An arrival is within the destination's time window.

Note that in all cases, a c-move trip is considered infeasible if it returns to the origin at the end of the day after the allowable time *TE*. In our implementation, all possible c-move trips are generated and checked for time window feasibility. Those that are infeasible are removed from the problem. Further, all shipments are checked to ensure that at least one c-move trip can cover them. If some shipments cannot be covered by any trips, they are flagged for manual intervention. If no intervention happens, they too are removed from the problem to ensure the feasibility of the model.

3. SOLUTION ALGORITHMS

We devise two solution algorithms for the CMO model: an exact column generation based approach and a heuristic greedy algorithm. During the preprocessing, we generate all possible c-move trips in the beginning and check them to ensure time window feasibility at origins, destinations, and on the roads (truck-bans). C-move trips comprising shipments beginning or ending outside of specified time windows are disregarded. Concurrently, we also check to ensure that at least one c-move trip exists to cover each and every shipment. If at the end there are some shipments that cannot be covered by any c-move trips, those shipments are removed from the problem to ensure feasibility of the model. In reality, each of those shipments will require manual interventions to ensure that at least one c-move trip can cover each of them. Those interventions may include time window modification at one or both ends, or fleet change (to improve truck speed), for example. Once the preprocessing is done, the model can be solved using one of the two solution algorithms.

3.1. Branch-and-Bound with Column Generation

Figure 9 summarizes our column generation based approach. It first removes a majority of the variables from the problem to form a restricted master problem (RMP), which is then solved to optimality using an ordinary branch-and-bound algorithm. The optimal solution obtained is checked whether it satisfies a pre-specified objective function value (a linear program lower bound in the CMO model). If it does, the optimal solution is obtained and the algorithm terminates. If it does not, the algorithm looks for variables that can improve the current solution. If new variables are found, they are added to the reduced-sized problem to form an expanded RMP, which is then resolved using the branch-and-bound algorithm again. From then on, the algorithm repeats until the bound is achieved or no new variables are found, at which point the optimal solution for the original problem is obtained.

3.2. Heuristics

The heuristic algorithm depicted in Figure 10 is a simple greedy algorithm. It starts by generating all feasible columns for the problem. Feasible columns are those that satisfy the time window requirements both at the origins and destinations as well as other side constraints such as maximum distance requirement. For each of the feasible column, we compute the *Trip Value Index* (*TVI*), the ratio between the total heavy haul distance of the trip and the total trip cost. TVI is a proxy for the value of the trip and can be used to rank relatively the efficiency of the c-move trip. Next, we sort all columns by TVI in a decreasing order (breaking ties arbitrarily). From here on the algorithm iterates until all shipments are covered. At each iteration, the algorithm selects a Trip *p* from the top of the sorted TVI List and check whether Trip *p* can be used to cover unsatisfied shipments. If yes, the algorithm checks the number of times Trip *p* is needed, marks Trip *p* as used, removes shipments satisfied by Trip *p* from further consideration, and removes Trip *p* from the TVI List. If no,

Trip p is simply removed from the TVI List. Then the algorithm checks whether all shipments have been satisfied. If no, the algorithm iterates by selecting the next trip from the TVI List. If yes, the algorithm terminates.

The algorithm terminates in finite iteration because we have ensured in the preprocessing step that all shipments are feasible before entering the solution algorithm.

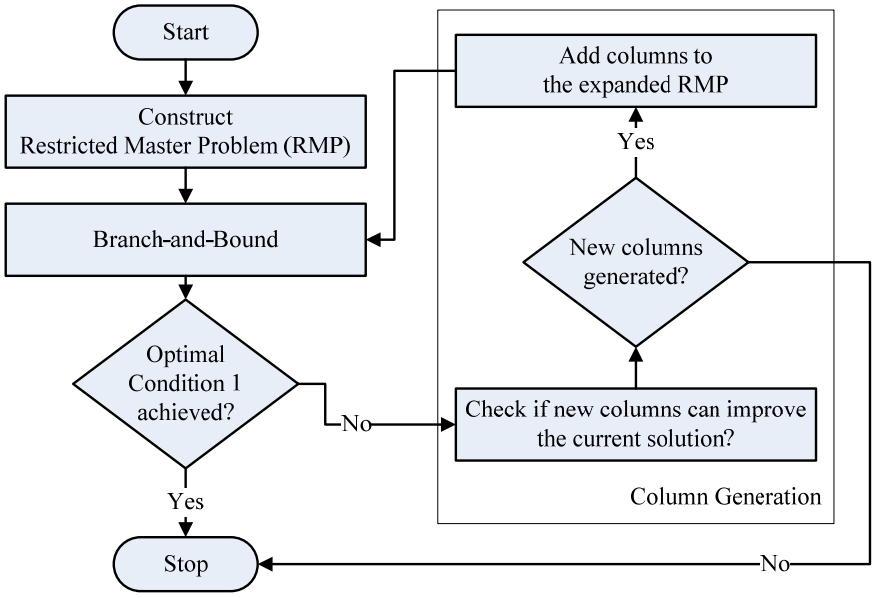

Figure 9 Simplified diagram of the branch-and-bound with column generation approach.

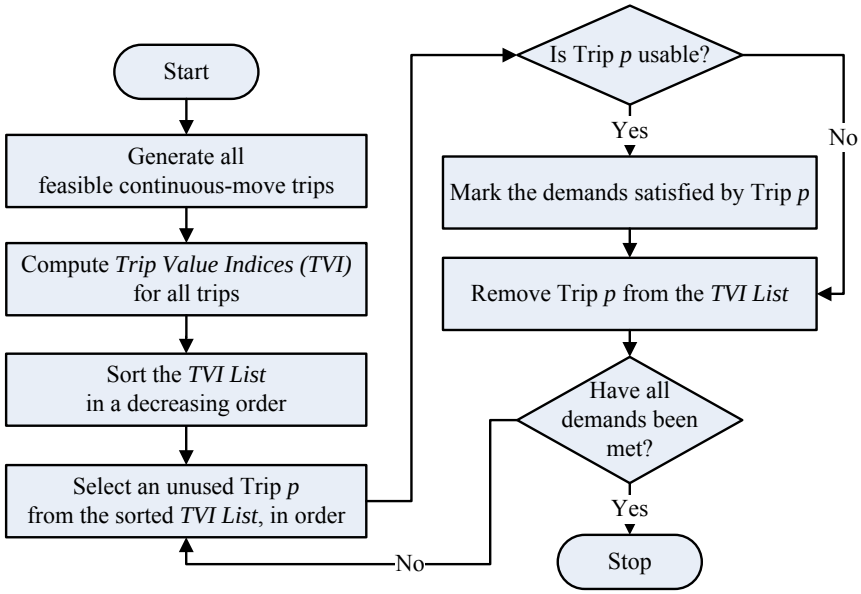

Figure 10 Heuristic algorithm

4. COMPUTATIONAL RESULTS

We test our model and algorithms using large-scale data derived from actual physical network distribution of a large logistics provider (Table 1). All tests are performed on a

Pentium IV 3.06 GHz desktop with 2 GB of memory. Three sets of runs—regular branch-and-bound on full-size problem, column generation, and heuristic—are performed and compared.

Table 1 Data characteristics

	Problem					
	1	2	3	4	5	6
No. of fleet types	6	6	6	6	6	6
No. of OD pairs	118	342	540	1,080	2,340	4,320
No. of shipments	314	1,058	1,984	3,960	9,539	13,657
No. of trucks at each origin	377	1,431	2,232	2,925	7,700	11,835
Origin time windows	8:00--18:00 (Return Trips before 21:00)					
Avg. destination time windows (hrs)	6.15	5.97	5.97	5.97	5.97	5.97
Total empty haul distance (km)	7,198	50,563	142,601	366,900	950,819	2,261,933

Figure 11 shows the resulting empty haul distances compared to a non-c-move operation. Our findings indicate that as problems become larger, empty haul reduction increases but at a decreasing rate because (i) more shipments allow for greater combination opportunities and (ii) this reduction slowly tends to the deadweight empty haul distance, which exists in most real life problems. Note that column generation algorithm gives high quality solutions comparable to those from using regular branch-and-bound algorithm, while the heuristic algorithm also gives substantial reductions in empty haul but not as large.

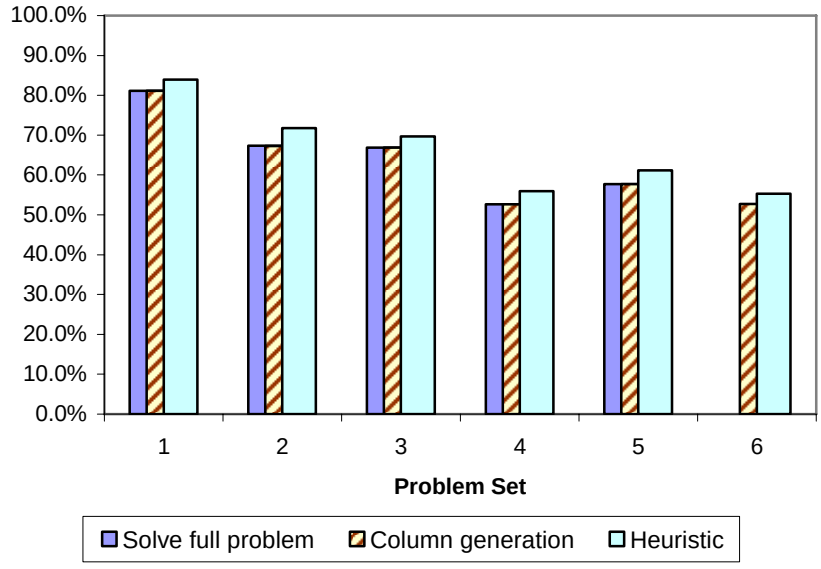

Figure 11 Empty haul distances (compared to non-c-move operation).

The heuristic algorithm, however, outperforms the other two algorithms in term of runtime (Figure 12). Column generation algorithm gives slightly longer runtimes compared to the heuristic while the regular branch-and-bound algorithm takes significantly longer.

Figure 13 shows the performance of our column generation algorithm. As the number of variables in the master problem increases, the smaller percentage of those columns is generated. In large problems with more than 1 million variables in the master problem, less than 1% of the columns are used in the final RMP.

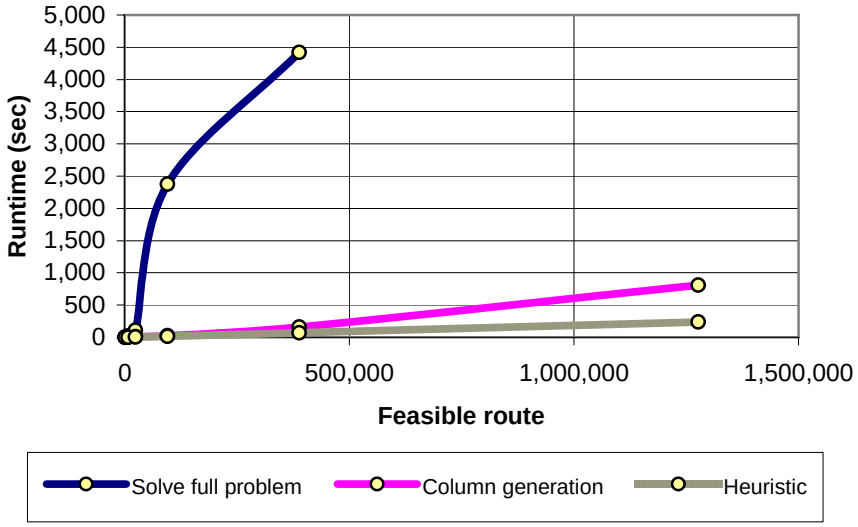

Figure 12 Runtimes.

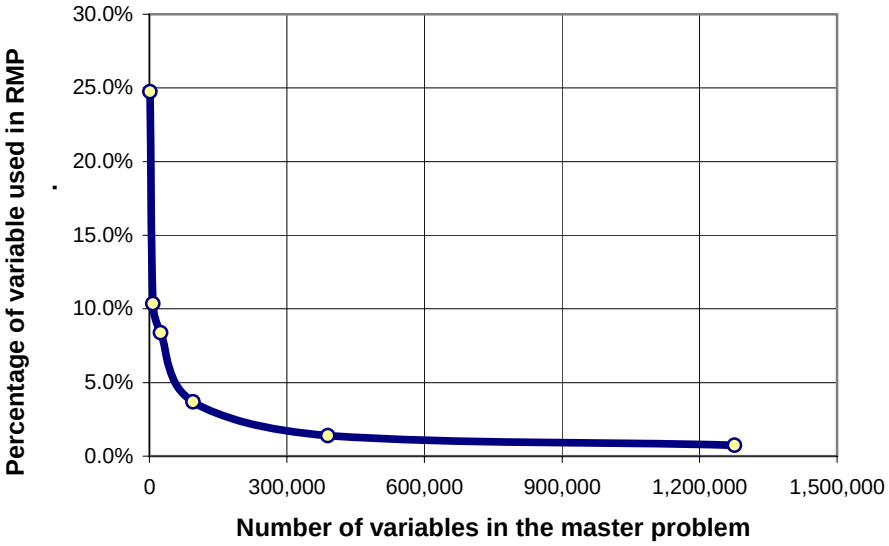

Figure 13 Percentage of variables used in the final RMP of column generation.

In summary, the comparison between the branch-and-bound with column generation approach and the heuristic shows slightly better cost savings with the former and markedly better runtimes with the latter. To decide which approach to use, it is a tradeoff between higher savings with the column generation approach and better runtimes as well as simpler implementation with the heuristic approach.

REFERENCE

Ergun, O., G. Kuyzu, and M.W.P. Savelsbergh (2005) Reducing truckload transportation costs through collaboration. To appear in *Transportation Science*.