Continuous Move Optimization for Truckload Pickup and Delivery Problem

Manoj Lohatepanont, Sc.D. (LManoj@alum.mit.edu)
Yossiri Adulyasak (a.yossiri@gmail.com)
Department of Civil Engineering, Chulalongkorn University

Abstract

We present in this paper a prototype optimization model for generating continuous move plans for the large-scale truckload pickup and delivery problem. Our model matches truckload moves for a given set of origin-destination (OD) pairs, allowing continuous moves of trucks (i.e., multiple and sequential pickups and deliveries) when possible. We present a set partitioning formulation and describe a solution algorithm based on the column generation technique. We test the model on large-scale problems with more than one million variables. Computational results show impressive runtimes and significant reduction of empty-haul distances, which translates to major saving in fuel and other operating costs as well as reduction in pollution released to the environment.

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

1. Introduction

Truckload (TL) transportation is a mode of land freight transportation, in which each truck picks up goods from an origin, transports, and delivers all goods to a destination without mid-route pickups or deliveries. From a given set of origin and destination (O-D) pairs, transport planners determine first the best route for each O-D pair, and later assign trucks to these predetermined routes. The problem of determining the best assignment of trucks to O-D routes is referred to as *the Truckload Pickup and Delivery Problem (TPDP)* and is typically cast as an assignment problem, where trucks are assigned to routes or lanes such that all lanes are covered and transportation costs are minimized (or other objectives achieved).

With each *head haul* move of the truck, goods are transported from its origin to its destination and revenue is generated. Without goods, the truck moves an *empty haul*, in which only costs are incurred and no revenue generated. If attempts to secure a transportation order from a destination location back to the location where that truck originates are unsuccessful, the truck will run an empty haul. These empty hauls represent a serious problem for truckload carriers, the trucking industry, as well as the country's economic system, as each empty haul does not generate any economic values.

The scale of the problem in Thailand is of serious concern. According to the statistics compiled by the Department of Land Transport [7], almost 690,000 trucks in the kingdom account for more than 89% of nationwide freight movements. Together they make over 71.7 million trips, covering more than 12,145 million kilometers and consuming over 3,470 million liters of fuel annually. It is projected that 46% of truck movements in the kingdom are empty haul moves, which equates to 33 million trips or 5,587 million kilometers of empty haul moves, translating to over 1,596 million liters of fuel or 22,538 million baht lost per year. This is a major economic loss for the country, especially in the current situation where fuel prices have skyrocketed. The Department of Land Transport's statistics also note that over 160,000 tons of pollution is released to the environment directly as a result of empty haul moves. The Department of Land Transport estimated that a decrease of 1% in empty haul moves will lead to a total monetary saving of over 720 million baht and a pollution reduction of 2,488 tons per year. Thus, empty hauls are a serious problem which need immediate attention.

A few remedies to the empty haul problem exist, e.g., back haul moves and continuous moves. In *back haul moves*, carriers secure goods for transporting back to the initial origin, thus making the back haul move a revenue and value generating move. In *continuous moves*, attempts are made to match multiple truckload pickups and deliveries to one truck in sequential order such that the prior delivery is made before the next pickup in the sequence. The benefit of continuous moves derives from the overall reduction in empty haul distances. Careful planning can ensure that the relocation of a truck from the prior delivery location to the next pickup location will minimize the overall empty haul distances for the entire network.

In this paper, we focus our attention on finding optimal routes for the *continuous move problem*, using a large-scale mathematical model. We employ the set partitioning formulation and devise a solution algorithm based on the column generation concept to solve quickly large-scale problems with over a million variables.

1.1. Contributions

Our contributions include:

1. apply advanced operations research techniques to solve the large-scale continuous move plan problem,
2. design specifically for the problem, a column generation based solution algorithm, and
3. implement and test the model using large-scale data sets.

1.2. Paper Organization

We give the formal description of the problem in Section 2. In Section 3, we review related literature and present a set partitioning model for constructing continuous move plans for large-scale truckload pickup and delivery networks. A solution algorithm based on the column generation technique is later presented in Section 3. Our implementation details and test results are shown in Section 4. Sections 5 and 6 contain discussions and conclusions of the findings and insights.

2. Problem Description

In this section, we begin with a detailed description of the problem. Next, we formalize the problem statement and detail our assumptions.

2.1. Definitions and Terminologies

A *truckload trip*, t^{ij}, is defined by its origin $i \in I$ and destination $j \in J$, where I and J are sets of origins and destinations respectively. The exact route that a driver may take to deliver goods from i to j is predetermined. This can be done using any shortest path algorithm to achieve a pre-specified goal of minimizing distance, time, fuel consumption, or any combination thereof. Each truckload trip is followed immediately by an empty haul move back to its origin. For each truckload trip t^{ij}, we compute the trip cost, c^{ij}, which may include fuel, driver, maintenance, depreciation costs, and other relevant cost items for the entire journey, including the empty haul return trip. (Note that we can easily change the objective of our model from cost minimizing to other objectives by changing the terms associated with each truckload trip here. For details see the discussion in Section 5.)

A *continuous move (c-move) trip*, p, occurs when two or more truckload trips are sequentially combined. That is, if trips $t^{i_1j_1}$ and $t^{i_2j_2}$ are combined, a c-move trip p will require a driver to (i) deliver goods from origin i_1 to destination j_1, (ii) make empty haul move to a new origin i_2, (iii) pick up goods from origin i_2 and deliver them to a final destination j_2, and (iv) return to the initial origin i_1. The exact route a driver may take from the prior destination to the new origin is predetermined in a similar manner to that of the head haul truckload trip. For each c-move trip p, we compute its cost, c_p, which includes (i) the summation of the costs (*excluding those associated with the empty haul return trips*) of all truckload trips included in p, (ii) the costs of empty haul relocating trips from the prior destinations to the new origins, (iii) the costs of empty haul return trip from the final destination of the day back to the initial origin of the day, and (iv) other c-move specific costs that may exist.

2.2. Assumptions

We assume the following in this paper:

1. daily operation,
2. deterministic truckload trip demand,

3. no delivery time requirement, and
4. fully combinable truckload trips.

We only consider a daily operation, in which all trips are planned for one day of operation in order to enforce and simplify truck location requirements, which dictate that all trucks beginning the day of operation at an origin i must return to that origin by the end of the day. The second assumption excludes stochastic and dynamic considerations. This is justifiable as the model that we propose is meant as a planning tool, not as an operational tool. The third and fourth assumptions allow us to consider more alternatives in creating c-move trips. It is easy, however, to relax these two assumptions as detailed in Section 5.

2.3. Problem Statement

The *Continuous Move Planning Problem* can be defined as follows:

Given a truckload distribution network comprising of origins, destinations, predetermined routes connecting the origins to/from the destinations along with the costs associated with each route, and truckload trip demands over the network, find the cost minimizing continuous move plan satisfying all truckload trips demanded.

3. The Model and Solution Algorithm

In this section, we describe our model in details. We begin with relevant literature review. Next, we move on to explain the notations, present the formulation, discuss its characteristics, and describe the solution algorithm.

3.1. Relevant Literature

There are two major approaches to solving vehicle routing problems: heuristics and mathematical programming. Heuristics are often employed to solve large-scale vehicle routing problem because of their ease of use and quick solution time compared to those of mathematical programming approaches. There are many heuristics one can use to suit specific problem specifications. Examples include: Clark and Wright [5], Kinderwater and Savelsbergh [9], and Renaud and Boctor [11]. Even though heuristics are easier to use, their drawbacks are often the inconsistent quality of the solutions.

Recent advancements in computing technology have enabled researchers to build and solve many previously intractable large-scale optimization problems. The vehicle routing problem also receives a lot of attention due to its wide applicability. Many models have been proposed for different variations of the problem. See, for examples, Fisher [8], Toth and Vigo [12], and Bramel and Simchi-Levi [4]. A majority of the literature in this area focus on ways to reduce the complexity of the problem or improve the solution time by introducing various techniques such as cut generations (to improve the effectiveness of the search in the branch-and-bound tree), column generations (to reduce the size of the problem), or both column and cut generations, in which case the full scale implementation becomes branch-and-price-and-cut, which is highly complicated and has not been shown to solve effectively large-scale problems.

In this paper, we introduce a mathematical programming formulation for solving a vehicle routing problem, but do not rely on the traditional solution approaches. Instead, we introduce a heuristic solution algorithm that guarantees optimality within a finite number of iterations. The remainder of this section details our approach.

3.2. Notations

We summarize and present all notations used below to facilitate our discussion of the model in the next subsection.

<u>Variables</u>

x_p	is the number of times a c-move trip p is used, or, equivalently, the number of trucks making the c-move trip p.
n^i	is the number of trucks beginning the day at an origin i.

<u>Parameters</u>

c_p	is the cost of a c-move trip p.
r_i	is the daily cost of using a truck from an origin i.
δ_p^{ij}	is the number of times a c-move trip p covers a truckload trip t^{ij}.
α_p^i	equals 1 if a c-move trip p begins the day at an origin i, 0 otherwise.
T^{ij}	is the number of demanded truckload trips from i to j.

N^i	is the number of trucks available at the beginning of the day at an origin *i*.

Sets

P	is the set of c-move trips, indexed by *p*.
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*.

Note that in order to simplify the notations used, a single truckload trip is defined as an elementary c-move trip using the same notation *p* (comprising of a single truckload trip).

3.3 Model Formulation

Our *Continuous Move Planning (CMP) Model* is:

$$Min \quad \sum_{p\in P} c_p x_p + \sum_{i\in I} r^i n^i \qquad (1)$$

Subject to:

$$\sum_{p\in P} \delta_p^{ij} x_p = T^{ij} \quad \forall ij \in O \qquad (2)$$

$$\sum_{p\in P} \alpha_p^i x_p - n^i = 0 \quad \forall i \in I \qquad (3)$$

$$x_p \geq 0 \quad \text{and integer} \qquad (4)$$

$$0 \leq n^i \leq N^i \quad \text{and integer.} \qquad (5)$$

3.4 Model Characteristics

Notice that the CMP model presented in Formulation (1)-(5) takes the form of the set partitioning model, in which the truckload trips between a given O-D pair have to be covered the exact required number of times (T^{ij}) by the optimal combination of c-move trips ($\sum_{p\in P} \delta_p^{ij} x_p$). Modeling the Continuous Move Planning Problem using the set partitioning formulation has a potential drawback in that the number of possible c-move trips grows exponentially with respect to the number of O-D truckload trips. In the extreme case, one has to enumerate all possible combinations of the required O-D truckload trips to get the complete set of possible c-move trips. Consequently, the model's size and complexity become intractable, and brute force attempts to solve these large-scale problems can be computationally prohibitively expensive.

We employ three column elimination measures, exploiting certain problem characteristics, to help eliminate infeasible and redundant columns in our CMP model. These include:

1. removing c-move trips with total distance exceeding the daily maximum allowable distance,
2. removing c-move trips containing more O-D truckload trips in a single c-move trip than demanded for that O-D, and
3. removing redundant c-move trips originating from the same origin and containing the same truckload trips but in different order.

The first and second measures aim to exclude infeasible c-move trips from the model. The third measure is used to remove redundant columns. Note that if delivery time assumption is relaxed, we need to modify the third measure to ensure delivery time requirements are met, which may positively or negatively affect the effectiveness of the measure depending on the requirements put in place. Section 4 shows the combined effectiveness of these measures in our computational tests. Lastly, note that these measures do not affect optimal solution quality as they do not completely remove any feasible solutions from the problem.

3.5 Solution Algorithm

Despite the column elimination measures described in the previous subsection, the total number of columns in the CMP model is still very large for reasonable size, real life problems (see Section 4). It is, therefore, necessary to employ an advanced solution algorithm to solve the CMP model. In particular, we devise a solution algorithm based on the *column generation* concept.

Column generation is a methodology for solving large-scale *linear programs (LP)*, especially those with a large number of columns. It first removes a majority of the columns from the original problem (called *master problem* or *MP*) to form a *restricted master problem* (*RMP*). The necessary requirement for the columns to be included in the RMP is that they are sufficient to form a basic feasible solution. Once a proper RMP is constructed, the first iteration begins by solving the RMP to optimality. Next, shadow prices from the optimal RMP's solution are used to compute or "*price*" the reduced costs of the excluded columns. If there exist columns with reduced costs that can improve the solution, those columns are added to the RMP. A new iteration begins. This process continues until no more columns are generated, at which point the algorithm terminates with optimal solution to the original Master Problem. There are many variations of column generation implementations. Interested readers are referred to Ahuja, Magnanti, and Orlin [1] or other advanced operations research texts for a proof and a more thorough review of the technique.

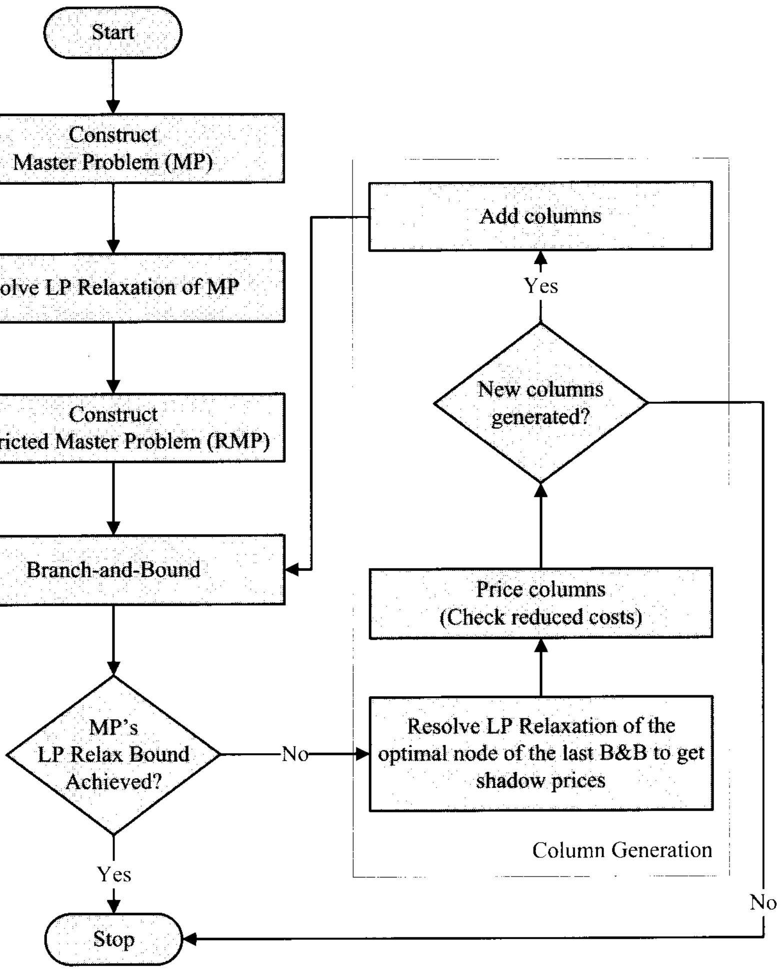
Start
Construct Master Problem (MP)
Solve LP Relaxation of MP
Construct Restricted Master Problem (RMP)
Branch-and-Bound
MP's LP Relax Bound Achieved?
No
Yes
Stop
Add columns
Yes
New columns generated?
Price columns (Check reduced costs)
Resolve LP Relaxation of the optimal node of the last B&B to get shadow prices
Column Generation
No

Figure 1: Solution Algorithm for the CMP Model

When column generation is applied to integer programs (IP), a much more involved process, called *branch-and-price*, in which columns are generated inside the branch-and-bound tree, is used. The problem is made much worse in large-scale problems (Barnhart, et al. [2]). In this paper, we devise an effective heuristic for solving large-scale integer program with column generation. Figure 1 depicts our algorithm. The algorithm calls for column generation outside of the branch-and-bound tree, thus, simplifying the implementation efforts and computational requirements.

We first construct the full master problem (MP), in which all columns are explicitly included. (Column elimination measures may be employed to reduce the MP's size.) Next, we solve the MP's linear program relaxation (LP relaxation), in which all integer requirements are relaxed to obtain an LP lower bound of the problem. We next construct the restricted master problem (RMP) by including (i) elementary c-move trips and (ii) all non-zero columns from the MP's LP relaxation. This proves to be an effective starting point for the CMP model. We now attempt to solve the RMP for the first time in branch-and-bound to get an IP optimal solution. The optimal solution obtained is optimal with respect to the RMP, but may not be optimal for the original MP because a great majority of the columns have not been considered.

Next the algorithm checks whether an optimality condition—an acceptable gap between the IP solution obtained and the MP's LP relaxation lower bound—is achieved. If this pre-specified gap is achieved, the algorithm stops and the optimal solution is obtained. The economic justification for using this gap as an optimality condition is, that even though there may exist columns that can further improve the solution, we do not need to consider them because the current solution is sufficiently close to the lowest possible bound on the objective function value. The technical justification for using this gap as an optimality condition is, that from our computational experience the lower bound obtained from the CMP master problem's LP relaxation is a tight bound, i.e., it is reasonably close to the optimal integer solution of the CMP master problem. Padberg [10] notes that the LP relaxation to a special class of set partitioning problem that has a perfect 0-1 matrix produces a tight bound. Although our constraint matrix is not a perfect 0-1, our computational testing has shown a similar characteristic, that is, the CMP master problem's LP relaxation bound is tight.

If, however, this optimality condition is not satisfied, we enter the column generation process. The next step is to fix variables in the optimal solution of the RMP and resolve it as an LP to obtain dual information. In particular, we need dual prices or shadow prices to compute or "*price*" the reduced costs of the excluded columns. (Readers are referred to standard operations research textbooks (e.g., Bertsimas and Tsitsiklis [3]) for more information on computing reduced costs from shadow prices.) If negative reduced cost columns exist, it implies that the current solution might be improved by adding these negative reduced cost columns to the RMP. In our implementation, at each iteration of the column generation process, we explicitly price excluded columns and add altogether a pre-specified number (e.g., 5,000) of negative reduced costs columns when we find them.

If, in an iteration, negative reduced cost columns are found, they are generated, i.e. added to the RMP. Then the algorithm repeats by solving the *expanded RMP* (i.e., the RMP from the previous iteration with added columns) in branch-and-bound to obtain an optimal integer solution (with respect to this particular expanded RMP). The MP's LP-bound gap is then checked; if it is not satisfied, more columns are priced and added if necessary. The process continues by adding more columns at each iteration to the expanded RMP—without ever removing any columns from the expanded RMP. The column generation process ends when no negative reduced cost columns are found, at which point the algorithm terminates and the optimal integer solution to the original MP is found.

The algorithm *will terminate in a finite number of iterations* because (i) the MP's LP-bound gap is achieved, (ii) no negative reduced cost columns are found, or (iii) all columns are generated and the expanded RMP reverts back to the original MP, which can be solved using a simple branch-and-bound algorithm.

4. Computational Results

In this section, we present our computational testing. The implementation is done in Visual C++ on a PC with Pentium IV, 3.06 GHz, and 2 GB memory. We use the SYMPHONY framework and CLP linear programming solver (COIN-OR Projects [4]) as our callable library.

4.1. Data and Model Characteristics

Table 1 shows the different characteristics of our test data. Data Set 1 is obtained from an actual transportation service provider. It represents a small portion of their full operation. Data Sets 2-7 are derived from Data Set 1 using actual locations of origins and destinations in the provider's network. The number and locations of truckload trips are manually generated for testing purposes. Maximum allowable distance for c-move trips for Data Set 1 is 600 km, for Data Sets 2-3 is 800 km, and for Data Sets 4-7 is 1,000 km per day.

Due to confidentiality reasons, the cost function employed in the objective function for this exercise is not an actual cost function of the transportation provider, but a general cost function comprising of variable costs (e.g., fuel, wear and tear) and fixed costs (e.g., depreciation, per trip charge). It should be noted that the actual figures used in this cost function does not affect the validity of the test or results shown in this paper because all comparisons are done on the same cost basis. If this cost basis is changed, all results will change accordingly in a similar direction.

From Table 1, notice that as the size of the problem increases, the number of rows increases at a relatively linear rate, but the number of columns grows much more quickly at an exponential rate. Table 1 also shows the performance of the column elimination measures, which are relatively effective for small size problems, but for large size problems these measures are only able to reduce about 1% of the number of columns. By contrast, the RMPs constructed for use with our column generation based solution algorithm are much smaller than the original MPs. Note that for the same problem instance, there are the same numbers of rows in MP and RMP.

In the next two subsections we present results from solving the CMP model without and with column generation to demonstrate (i) the benefit of using the CMP model in constructing c-move plans, and (ii) the performance of our solution algorithm, respectively.

Table 1: Data and Model Characteristics.

	[illegible]					MP			RMP
Data Sets	No. of Orig.	No. of Dest.	No. of O-D Pairs	No. of Truckload Trips per Day	No. of Trucks at each Origin	No. of Cols (Before Col Elim.)	No. of Cols (After Col Elim.)	No. of Rows	No. of Cols
1	4	14	14	46	10	210	165	18	70
2	9	47	51	222	25	2,652	2,495	60	242
3	9	51	76	326	30	5,852	5,469	85	365
4	9	129	225	803	80	50,850	50,360	234	1,025
5	9	77	315	1,278	80	99,540	99,345	324	1,490
6	9	166	900	3,368	250	810,900	803,249	909	4,114
7	9	166	1,494	6,756	400	2,233,530	2,215,091	1,529	6,903

4.2. Solving CMP Model's Master Problem

Table 2 shows the results from solving the master problem without using column generation and compares them to the results from manual planning. It is difficult to construct reasonable manual plan for the purpose of comparison in this exercise as the quality of the manual plan depends very much on the planner's experience and we do not have the transportation provider's actual c-move plan for these data sets. The manual plans used, therefore, are simple O-D truckload trips, which give overestimates of the empty haul distances.

From Table 2, the operating cost saving range from 10% to over 25% and the reduction in empty haul distances ranges from 20% to almost 60%. The reason for variations in these savings derives from the differences in the underlying network structure and in the demand instance. Despite using overestimates of the empty haul distances in the manual plans, the results strongly indicate that significant savings can be gained from using an optimization model in the continuous move planning process, especially in large size problems where manual planning is certain to be far from optimal.

Although the cost function used in this test is not the actual cost function from the transportation provider, we take care in formulating our own cost function. The approximate average truck operating costs come out to be between 8 and 9.5 baht/km, a reasonable range for total operating costs per km. Notice now the scale of the operating cost savings from using the CMP model. The savings are on the order of hundreds of thousands of baht for small to medium size problems (Data Sets 1-3). For large size problems (Data Sets 4-5), the savings exceed 1 million baht per day, translating to over 250 million baht savings per year. The saving estimates are compared to the case where no c-move trips are manually planned, but even when taking that into account, one can clearly get the sense of the benefits of the CMP model.

Table 2: Results from solving the CMP's master problem compared to manual plans.

	Operating Cost (Baht)			Empty Haul (km)			MP Run Times (min)
Data Sets	Manual	Optimal	% Diff.	Manual	Optimal	% Diff.	
1	84,090	73,115	13	4,644	3,225	31	0.1
2	485,210	384,900	21	28,238	15,998	43	0.2
3	714,450	586,520	18	41,964	26,474	37	0.3
4	1,693,190	1,247,950	26	99,499	35,969	64	20.0
5	2,018,000	1,811,665	10	105,305	84,712	20	31.3
6	6,865,380	5,203,310	24	399,825	166,823	58	101.0
7	13,512,880	*	*	784,136	*	*	> 64 hrs

* No integer solution obtained after 64 hours.

In terms of runtime, the model solves relatively quickly—less than one minute, for small and medium size problems. For large size problems, the solver takes anywhere from 20 minutes to a few hours. For very large size problem (Data Set 7), however, the solver fails to reach an integer solution after 64 hours. Note that in all instances, the optimality gap of less than 0.5% is achieved.

4.3. Solving CMP Model using Column Generation

Results from column generation are shown in Table 3. The values in the "diff." column are differences measured against the results from the CMP model *without* column generation (Table 2). In all instances, the operating costs and empty haul distances from the CMP model with column generation are within less than 1% of those of the original CMP's master problems. That is, the quality of the solutions is comparable to that from solving the CMP's full size master problems without column generation.

The benefits of our column generation based solution algorithm as displayed in Table 3 are the significant runtime reduction (Data Sets 4-7) and the ability to attack very large, previously intractable problems (Data Set 7). In small problems, the saving in the runtimes is negligible because the column generation process is more complex, but for large problems, however, the runtime reductions become very significant (up to 98% reduction). Observe as well that we are now able to solve the largest problem a (Data Set 7) with over 2.2 million columns, which was previously unsolvable after 64+ hours, in approximately 80 minutes.

The benefits of using column generations derive from the ability to reach optimal or near optimal solutions by considering correctly only fractions of the total number of columns. Table 4 shows different statistics of the test. All instances were terminated after at most four iterations by achieving the optimality condition set by the master problem's LP relaxation lower bound (Table 4). Observe that as the problem grows, our solution algorithm considers proportionally smaller sets of columns (from 25% for Data Set 1 to only about 1% for Data Set 7).

Table 3: Results from solving the CMP model using column generation.

			Empty Haul (km)		Run Times (Min)		
Data Sets	Col. Gen. Optimal	% Diff.	Col. Gen. Optimal	% Diff.	Col. Gen.	Diff.	% Saving
1	73,115	0.00%	3,225	0.00%	0.1	0.0	0%
2	385,030	-0.03%	16,016	-0.11%	0.2	0.0	0%
3	586,650	-0.02%	26,470	0.02%	0.3	0.0	0%
4	1,248,860	-0.07%	36,056	-0.24%	0.4	19.6	98%
5	1,812,195	-0.03%	84,750	-0.04%	2.5	28.8	92%
6	5,204,655	-0.03%	166,486	0.20%	20.5	80.5	80%
7	10,401,490	*	349,350	*	80.5	*	*

* No integer solution obtained after 64 hours.

Table 4: Column Generation Statistics

Data Sets	No. of Cols in the MP	No. of Cols in the Final Expanded RMP	% of Cols Considered	No. of Col. Gen. Iterations	MP's LP Relax Gap
1	165	41	25%	0	0.00%
2	2,495	477	19%	4	0.05%
3	5,469	857	16%	4	0.04%
4	50,360	3,976	8%	4	0.12%
5	99,345	4,405	4%	2	0.00%
6	803,249	21,933	3%	4	0.03%
7	2,215,091	28,770	1%	3	0.03%

5. Further Discussion

We now revisit the model's assumptions detailed in Section 2 to discuss means to relax them. The daily operation assumption is easy to relax. The potential pitfall of considering more than one day of operation is in losing the tractability of the problem quickly as the c-move trips are extended. The deterministic truckload trip demand assumption makes sense for planning purposes, but if we were to use this model in operational settings, where demands are stochastic and dynamic, additional research would be needed to develop a stochastic and/or dynamic model. The assumptions on delivery time requirements and fully combinable trips can be relaxed readily. If there are delivery time requirements, we can easily capture those requirements in the way we construct our c-move trips (columns). In fact, with those requirements, our choices will potentially be more limited and hence the number of columns will be reduced, making the problem even smaller. Similarly, introducing restrictions on which trips cannot be combined with other trips, will limit our choices, and hence reduce the number of columns, and ultimately the model's size.

Another element that is worth mentioning is the objective function. In our test, we assume a cost minimization model, but if the goal of the transportation provider is different, the objective function can easily be modified to capture complicated revenue/cost/profit functions.

6. Conclusion

Even though empty haul moves arise naturally from the imbalance of trade or freight movements within the transportation network, they are a serious problem for Thailand, as the economic figures in Section 1 show. In this paper, we focus on a transportation management solution to the problem. Specifically, by intelligently combining two or more truckload trips into a sequential continuous move trip, we can reduce the empty haul distances. Applying this seemingly simple concept to the entire network can lead to significant overall savings, but as the network size grows, the problem quickly become intractable as the sheer size of all possible combinations of trips become prohibitively large. Due to the size and complexity issues, a special solution algorithm is needed. We develop and test the CMP model using our column generation based solution algorithm in this paper. We demonstrate that our approach can be applied to very large scale problems without losing tractability.

The actual monetary benefits of our approach clearly depends on the scale of the operation of the transportation provider and the demands, but our test have shown that an operating cost saving between 10% and 25% is achievable with our CMP model. Our ballpark calculation translates this to over 250 million baht per year for a large size operator with 1,000 to 3,000 truckload shipments per day. Apart from the monetary benefits, the reduced empty haul distances translate to less fuel wasted (positively impacting the overall economy of the country) and consequently less pollution released to the environment.

7. References

[1] Ahuja, R.K., T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, New York, 1993.

[2] Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsgergh, and P.H. Vance, Branch-and-Price: Column Generation for Solving Huge Integer Programs, *Operations Research,* Vol. 46, #3, pp. 316-329,1998.

[3] Bertsimas, D. and J. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, New York, 1997.

[4] Bramel, J. and D. Simchi-Levi, On the Effectiveness of Set Covering Formulations for the Vehicle Routing Problem with Time Windows, *Technical Report*, Dept of Industrial Engineering and Management Science, Northwestern University, 1998.

[5] Clarke, G. and J. Wright, Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research*, Vol. 12, #4, pp. 568-581, 1964.

[6] COIN-OR Projects, http://www.coin-or.org/projects.html, 2006.

[7] Department of Land Transport, The Initiative to Reduce Energy Lost from Empty Haul Truck Moves, *Energy Saving Best Practices in the Transport Sector*, Council of Engineers of Thailand, Bangkok, 2006.

[8] Fisher, M.L., Optimal Solution of Vehicle Routing Problems Using Minimum K-trees, *Operations Research*, Vol. 42, pp. 626-642, 1994.

[9] Kinderwater, G.A.P. and M.W.P. Savelsbergh, Vehicle Routing: Handling Edge Exchanges, *Local Search in Combinatorial Optimization,* Wiley, Chichester, 1997 .

[10] Padberg, M.W., Perfect Zero-one Matrices, *Mathematical Programming*, Vol.6, pp. 180-196, 1974.

[11] Renaud, J. and F.F. Boctor, A Sweep-Based Algorithm for the Fleet Size and Mix Vehicle Routing Problem, *European Journal of Operational Research* 140, pp. 618-628, 2002.

[12] Toth, P. and D. Vigo, Branch-and-bound Algorithms for the Capacitated VRP, *The Vehicle Routing Problem*, SIAM: Philadelphia, pp. 29-52, 2001.