ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 783-788
doi: 10.18178/wcse.2019.06.116

Network Design for Express Package Delivery Service of Electric Vehicles

Xinxin Shi [1+], Hong Ma [2]

[1,2] School of Management, Zhejiang University

Abstract. With people's enhancement of ecological consciousness, electric vehicles should be considered into network design for express package delivery service. This paper proposes a cycle-based model for this new problem and constructs an efficient heuristic algorithm to solve large-scale problem. The heuristic algorithm combines tabu search metaheuristic method and column generation based heuristic method. Intensification strategy plays a key role on finding high quality solutions. The algorithm performs well on small size instances compared with branch and price algorithm and it can handle large-scale network efficiently.

Keywords: network design, electric vehicle, tabu search, column generation.

1. Introduction

In China, the demands for express package delivery service are increasing progressively in recent decades due to flourishing online shopping. About 50 billion packages were delivered throughout this country in 2018. Battery electric vehicles will become a necessary option to support these delivery services between cities in order to reduce their carbon footprint. Therefore, how to schedule these electric vehicles to pick up and delivery express packages between cities and deciding when and where to recharge themselves to minimize total system costs form the problem.

In our problem, given the information of demands for delivery service (origin, destination, quantity, time window) and a finite number of electric ground vehicles, we need to route package flows and also route electric vehicles to transport all the packages. Package flow of each demand can be split and may use more than one path from origin to destination. Such package flows can also be unloaded from current vehicles and reloaded to other vehicles later at certain terminal like [1] (Fig.1). Vehicles are required to come back to their original terminals after one time period and visit recharging stations when needed [2].

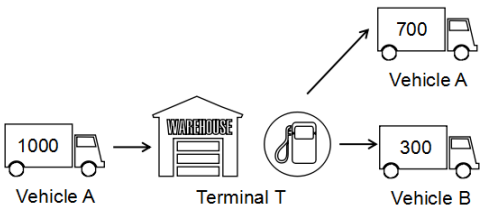

Fig. 1: A demand with a quantity of 1000 is split into two parts at terminal T. The package flow of 300 is transferred from vehicle A to B

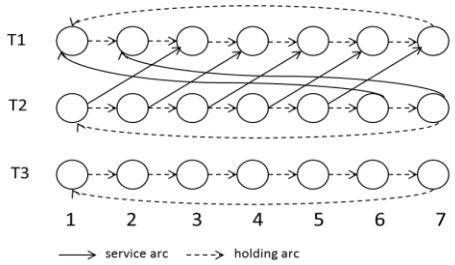

Fig. 2: An example of a time-space network with 3 terminals and 7 periods.

+ Corresponding author. Tel.: + 8618868102751
E-mail address: xxshi@zju.edu.cn

In this paper, we present a cycle-based mathematical formulation based on time-apace network firstly. Then we propose a tabu search metaheuristic algorithm which combines column generation based heuristic method to solve the problem. The performance of this algorithm would be evaluated in experimental section.

2. Mathematical Formulation and Column Generation

2.1. Mathematical formulation

Our formulation is built on a time-space network [3] (see Fig.2) in which each terminal and each arc has a copy in each period of scheduling horizon. Holding arcs represent that vehicles or packages wait at certain terminals and service arcs represent actual movement between two terminals.

In this formulation, $\tau \in \theta_{pl}$ represents a possible cycle-based route for an electric vehicle originating from terminal l and with capacity index p, and $\theta_p = \sum_{l \in L} \theta_{pl}$, $\theta = \sum_{p \in P} \sum_{l \in L} \theta_{pl}$. x_{ij}^k and z_τ are continuous and integer decision variables respectively, which determine the route of express packages and electric vehicles. The objective function minimizes sum of three types of costs: service operating cost of vehicles, fixed cost of vehicles and commodity cost. v_{ij}, f_{pl} and c_{ij} are parameters of these costs. r_{ij}^τ equals 1 if route τ covers arc (i, j), and equals 0 otherwise. Constraint (1) corresponds to flow conservation equation. w^k is the quantity of demand k and o(k), d(k) are its origin and destination. Constraint (2) ensures that on each service arc, total quantity of packages flowing over cannot exceed total capacities of vehicles supporting this arc. C_p is the capacity of vehicles with capacity index p. Constraint (3), named resource bound constraint, limits the number (ub_{pl}) of vehicles available at each terminal l and with capacity index p.

LP relaxation of above model will be solved by column generation. Next section will discuss how to identify feasible and good routes of electric vehicles in subproblems to optimize LP problem.

x_{ij}^k: amount of demand k flowing over arc (i, j)
z_τ: amount of cycle $\tau \in \theta$ selected

N: set of time-space network node
E: set of service arcs
H: set of holding arcs
A: set of all arcs
K: set of demands
P: set of capacities
L: set of physical terminals

$$\min \sum_{\tau \in \theta} \sum_{(i,j) \in E} v_{ij} r_{ij}^\tau z_\tau + \sum_{p \in P} \sum_{l \in L} \sum_{\tau \in \theta_{pl}} f_{pl} z_\tau + \sum_{k \in K} \sum_{(i,j) \in E} c_{ij} x_{ij}^k$$

$$s.t. \sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(j,i) \in A} x_{ji}^k = \begin{cases} w^k & i = o(k) \\ -w^k & i = d(k) \\ 0 & otherwise \end{cases} \quad \forall k \in K, i \in N \qquad (1)$$

$$\sum_{k \in K} x_{ij}^k \le \sum_{p \in P} \sum_{\tau \in \theta_p} C_p r_{ij}^\tau z_\tau \quad \forall (i,j) \in E \qquad (2)$$

$$\sum_{\tau \in \theta_{pl}} z_\tau \le ub_{pl} \quad \forall p \in P, l \in L \qquad (3)$$

$$x_{ij}^k \ge 0 \quad \forall k \in K, (i,j) \in A \qquad z_\tau \in N \quad \forall \tau \in \theta$$

2.2. Column generation

In column generation, master problem (MP) is the LP relaxation of original formulation containing subset of θ. For each capacity p and terminal l, there will be one corresponding subproblem to identify better vehicle routes which will be added to MP.

Since delivery services are offered by electric vehicles, we must take battery capacity into consideration, which means that an electric vehicle needs to recharge itself at any terminal before it is out of power. Here we consider full recharges only. We do not calculate recharging cost in objective function but it will take two hours for a single charge. *Capacity* denotes each vehicle's maximal capacity of battery power. And it is measured by the distance a vehicle can travel (In our problem, distances of all arcs are integers). Each time MP is solved, we update our subprolems and use dynamic programming to solve them. Let μ_a and π_{pl} be negative dual variables of constraints (2) and (3) in MP. For a subproblem with capacity p and original terminal l, the reduced cost of this subproblem becomes

$$\bar{c}_{pl} = \sum_{a \in E} v_a r_a^\tau + f_{pl} + \sum_{a \in E} C_p \mu_a r_a^\tau - \pi_{pl} = \sum_{a \in A} c_a^* r_a^\tau + f_{pl} - \pi_{pl}$$

where c_a^* is the updated cost of arc a. Dynamic programming algorithm for this subproblem is as follows.

Let T denote time period of our problem. A vehicle can start from terminal l at any time. So we should consider T different starting times. We denote starting node as l_t. Then for each starting time t, we should

find a shortest path from l_t to l_t after explicit one time period T, which must satisfy battery capacity requirement. Let f (n, w) record the shortest distance from l_t to n, when there are w units of electric power left for the vehicle. Then for each node n, we should consider two situations. One is that vehicle arrives at node n from other nodes. The other is that vehicle just finishes its charging at node n.

In the first situation, we enumerate all arcs pointing to n. For each arc a, we use d to represent travel distance and n_0 to represent head point of arc a. For each power capacity w_0 ($w_0 - d$ should be nonnegative) at n_0, we have

$$f(n, w_0 - d) = \min\{ f(n, w_0 - d), f(n_0, w_0) + c_a^* \}$$

In the second situation, let n_0 be the corresponding charging node in time-space network. Then we have

$$f(n, Capacity) = \min_{\forall w \in [0, Capacity]} f(n_0, w)$$

Notice that after we calculate all f (n, w) for node n, we will set some f (n, w_0) to be a very large positive value if there exists a w_1 larger than w_0 while f (n, w_1) is smaller than f (n, w_0).

In initialization phase, we set all f (n, w) to be very large positive values and f (l_t, Capacity) to be 0. Dynamic programming will enumerate times and then terminals for these times in time-space network to update our f (n, w), until we find the shortest feasible path. And we will add such path to MP when its corresponding reduced cost is negative.

3. A Tabu Search Heuristic Algorithm

In this section, we propose a tabu search heuristic algorithm in order to find high quality solutions within a limited time. We illustrate the steps of our approach in Fig.3.

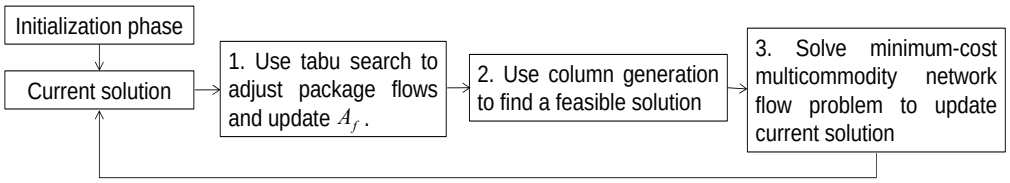

Fig. 3: Steps of heuristic algorithm.

In initialization phase, we first use Yen's algorithm [4] to find λ shortest paths for each demand k. For time-space network, there may be several arcs between two terminals with different periods all belong to k 's shortest paths. In that case, we tend to pick up the arc appearing more times in other demands' λ shortest paths. All the arcs we pick up form a refined network which can ensure the feasibility of all demands. Then a column generation based heuristic method is used to solve the problem based on this refined network. This method first uses column generation to solve LP relaxation of the problem, and then keeps the cycles generated in MIP, which will be solved by a commercial MIP solver. Since we usually set adequate electric vehicles for each terminal, this method can always help us find a feasible solution in our experiments.

3.1. Use tabu search to adjust package flows

Tabu search aims at re-directing package flows and updating subset of arcs, A_f, dynamically. Based on current solution, we will generate a neighbourhood flow distribution for each node $n \in N$ in time-space network. Then we choose the best one we evaluate and update A_f. This process is as follows (Algorithm 1).

1) Pick up and remove all package flow paths related to current node n

Our cycle-based neighbourhood procedure [5] first finds all the arcs (i, j) ∈ A pointing to or from node n. For each demand k flowing over arc (i, j), we use breadth-first search to explore a path as far as we can, which is the Source_Search algorithm in [6]. Afterwards we remove all these flow paths and then remove empty vehicle service arcs from current solution.

To evaluate total removed costs, at the beginning of tabu search we remove all empty vehicle service arcs and evaluate average fixed vehicle using cost of unit distance for each capacity as $\bar{f}_p$. Then the removed

costs are evaluated by the sum of package flow cost, operating cost of removed empty vehicle service arc and average fixed cost of vehicles, which is denoted by $cost_{remove}$.

2) Generate residual network for each package flow path and redirect them in the network

To redirect removed flow paths, residual network is generated for each of them. In a residual network, cost of each holding arc is 0. For one flow path with quantity x, if the residual vehicle capacity is larger than x on a service arc (i, j) ∈ E, the cost of this arc will be $c_{(i,j)} = c_{ij}x$. Otherwise, we decide if we should add another vehicle service arc or modify current vehicle service arc's capacity. Then we define the cost as $c_{(i,j)} = \min\{v_{ij} + \bar{f}_p, \bar{f}_{\Delta p}\} + c_{ij}x$. In both situations, we choose the least capacity p as long as it can support the amount of x commodities to flow over.

Algorithm 1 Step 1 Use tabu search to adjust package flows

1: **for** each node $n \in N$ **do**
2: $s \leftarrow$ current solution
3: Remove all empty vehicle service arcs and calculate $\bar{f}_p$;
4: Apply $Source\_Search$ to n and add all package flow paths to $pathSet$;
5: Remove all flow paths in $pathSet$ from s;
6: Remove all empty vehicle service arcs from s and calculate $cost_{remove}$;
7: **for** each $flowPath \in pathSet$ **do**
8: Generate residual network for $flowPath$;
9: Find shortest path to redirect $flowPath$ and adjust $cost_{redirect}$;
10: Update information of vehicle service arcs and flow distribution;
11: **end for**
12: $cost_{total} \leftarrow cost_{remove} + cost_{redirect}$
13: **end for**
14: Choose the neighbourhood with minimal $cost_{total}$ and update A_f

Notice that in residual network, we will set costs of arcs pointing to or from node n and also nodes in tabu list to be very large positive values to forbid flows running on these arcs. In addition, we add negative adjusted cost to some "key" service arcs which we surmise express packages flow on them would lead to better results. When there are packages flowing over a service arc (i, j) ∈ E, or a service arc (i_0, j_0) ∈ E satisfies $j_0 = i$ or $i_0 = j$, we consider such arcs as key service arcs.

Finally, we find the shortest path from the flow path's origin to destination in this residual network and redirect the flow. Then we update the vehicle service arcs information and flow distribution for next flow path redirection. We evaluate the cost of this step as the sum of all shortest paths' costs and denote it as $cost_{redirect}$. Total cost of this neighbourhood is $cost_{total} = cost_{remove} + cost_{redirect}$.

After the above two steps, we choose the neighbourhood with minimal $cost_{total}$. The move modifies the arc set all package flows cover and we update it as A_f. Corresponding node n will be added to tabu list.

3.2. Column generation based heuristic method

In this step, we limit that all packages can only flow over arcs in A_f. Cg-based heuristic method we mentioned in initialization phase will be used to get a feasible solution s. Note that this method might not lead to feasible solutions. In this case, we will round up the LP solution to integers to get a feasible solution.

3.3. Minimum-cost multicommodity network flow problem

In step 3, we fix the routes of vehicles in solution s. And we solve a minimum-cost multicommodity network problem in this capacitated network to improve solution s and then go to the next iteration.

Intensification

In intensification phase, we use commercial MIP solver to solve the problem with subset of vehicle route variables. In each iteration, we pick up all vehicle route variables from current solution and add them to the solver. The solver will try to find high-quality solution within a limited time. If some vehicle route variables have not been used in intensification solution for many iterations, we will remove them from the solver.

Note that this algorithm terminates when maximum unimproved iteration, $IterLimit$, is reached.

4. Computational Results

We test our algorithm on 5 small and 5 large size data sets. Small data sets use the information of network structure, time period and commodity information of small instances in [7]. large date sets are generated by ourselves which aim to explore the scale of problem this algorithm could handle. Parameter setting is based on what was observed to yield reasonable performance in calibration phase.

For small data sets, we compare our algorithm with a branch and price (B&P) method. Two kinds of capacity are considered. The parameters and results are shown in Table 1.

Table 1: Computational results for small instances

Instance	$\|L\|$	$\|S\|$	T	$\|K\|$	LB	B&P UB	B&P time	MIP solution	Gap	time
1	5	10	15	20	3836	3836	29s	3836	0%	2458s
2	5	15	20	25	3869	3869	4928s	3869	0%	2103s
3	5	15	25	25	3639	3661	36000s	3675	0.99%	467s
4	5	15	15	100	15426	15426	1881s	15426	0%	2960s
5	5	15	15	200	32900	32900	139s	32900	0%	278s

In Table 1, $|L|$, $|S|$ and $|K|$ represent the number of terminals, physical arcs and delivery demands respectively. T is time period. LB and B&P UB are the lower bound and upper bound got by branch and price, respectively. Last three columns are best feasible solution, gap value and computation time of the proposed algorithm.

From the above table, we see that tabu search algorithm can find optimal solutions which are proved by B&P for 4 of 5 small instances. For instance 3, it is acceptable to spend about 8 minutes to get a solution with a gap of 0.99%. We can see from instance 1 and 4 that tabu search spends more time than B&P, which might be unexpected because heuristic method usually spends less time than exact algorithm. We believe there is a tradeoff between solution quality and computational time. This tabu search algorithm may not converge as quickly as other heuristic algorithms. But solution quality is improved to some extent. Therefore, we conclude that this tabu search algorithm can find good enough solutions for small instances compared with branch and price algorithm.

For large date sets, we consider 3 kinds of capacity: 20,50 and 100. The results are presented in Table 2.

Table 2: Computational results for large instances

Instance	$\|L\|$	$\|S\|$	T	$\|K\|$	Initial solution	MIP solution	Iteration	time
6	10	50	30	100	12042	10023	20	3842s
7	10	50	50	400	94238	74991	5	5538s
8	20	230	48	200	41138	38947	5	7123s
9	20	300	72	200	33477	28377	5	9345s
10	30	516	48	400	83864	75950	5	10846s

Table 2 shows that the proposed heuristic algorithm can successfully handle large-scale problems with at most 26208 arcs in a network. It can find a solution much better than the initial one. Step 2 in Fig. 3 usually costs most of the solving time. Note that for all instances, best solutions are always found during intensification phase. Flow adjustment of step 1 guides the algorithm to explore important vehicle variables.

5. Conclusion

We present a cycle-based formulation of network design for express package delivery service of electric vehicles. The experiments show that the tabu search heuristic algorithm works quite well for small instances. It also fits for large-scale network which is a challenge for existing research of service network design with design-balance constraints problems. Intensification phase in our algorithm plays a vital role on finding high quality solutions. And we believe the ideas in [8] will enhance the performance for larger size problems.

6. References

[1] Kerivin, H., Lacroix, M., Mahjoub, A. R., & Quilliot, A. (2008). The splittable pickup and delivery problem with reloads. European Journal of Industrial Engineering, 2(2), 112-133.

[2] Schneider, M., Stenger, A., & Goeke, D. (2014). The Electric Vehicle-Routing Problem with Time Windows and

Recharging Stations. Transportation Science, 48(4), 500-520.

[3] Crainic, T. G., Hewitt, M., Toulouse, M., & Vu, D. M. (2016). Service Network Design with Resource Constraints. Transportation Science, 50(4), 1380-1393.

[4] Yen, J. Y. (1971). Finding the K Shortest Loopless Paths in a Network. Management Science, 17(11), 712-716.

[5] Ghamlouche, I., Crainic, T. G., & Gendreau, M. (2003). Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. Operations Research, 51(4), 655-667.

[6] Vu, D. M. , Crainic, T. G. , & Toulouse, M. . (2013). A three-phase matheuristic for capacitated multi-commodity fixed-cost network design with design-balance constraints. Kluwer Academic Publishers.

[7] Andersen, J., Christiansen, M., Crainic, T. G., & Gronhaug, R. (2011). Branch and Price for Service Network Design with Asset Management Constraints. Transportation Science, 45(1), 33-49.

[8] Boland, N., Hewitt, M., Marshall, L., & Savelsbergh, M. W. (2017). The Continuous-Time Service Network Design Problem. Operations Research, 65(5), 1303-1321.