IntRoute: An Integer Programming Based Approach for Best Bus Route Discovery

Chang-Wei Sung[1], Xinghao Yang[2(✉)], Chung-Shou Liao[1], and Wei Liu[2]

[1] Department of Industrial Engineering, National Tsing Hua University, Hsinchu, Taiwan
sung103034036@gapp.nthu.edu.tw, csliao@ie.nthu.edu.tw

[2] School of Computer Science, University of Technology Sydney, Ultimo, Australia
xinghao.yang@student.uts.edu.au, wei.liu@uts.edu.au

Abstract. An efficient data-driven public transportation system can improve urban potency. In this research, we propose IntRoute, an Integer Programming (IP) based approach to optimize bus route planning. Specifically, IntRoute first contracts bus stops via clustering and then derives a new bus route via a mixed integer linear program (ILP). This two-phase strategy brings three major merits, i.e., a single bus route without any transfer, the minimal total time consuming, and an efficient optimization algorithm for large-scale problems. Experimental results show that our IntRoute significantly reduces the traditional commuting time in Sydney from 31.53 min down to 18.06 min on average.

1 Introduction

In this study, we consider an important data-driven public transportation problem: finding the best bus route that minimizes passengers' overall commuting time. Given a bus transportation system as well as the requests of specific passengers who commute from different starting locations to a fixed destination, the goal of the problem is designing a new bus route to satisfy the passengers' demand without any transfer.

Previous studies on bus transfer problems were mostly not data-driven [2] due to data skewness problems [3]. In this work, we proposed a new integer-programming based method, which we call IntRoute, to find the route that minimizes the total time cost of the targeted passengers. Specifically, our IntRoute contains two main phases, i.e., the contraction of bus stops via K-means clustering and the derivation of new bus route via a mixed integer linear programming (ILP). The major contributions of this research are listed below.

- We design a single bus route in which all passengers with an identical destination are delivered without any transfers.
- We present a two-phase framework to minimize the total time expense of the specific passengers.
- We develop a genetic algorithm (GA) to solve the integer linear programming (ILP) for large-scale instances.

C. S. Jensen et al. (Eds.): DASFAA 2021, LNCS 12683, pp. 604–607, 2021.
https://doi.org/10.1007/978-3-030-73200-4_44

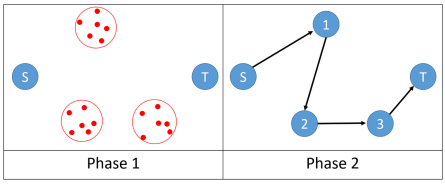

Fig. 1. The framework of the IntRoute.

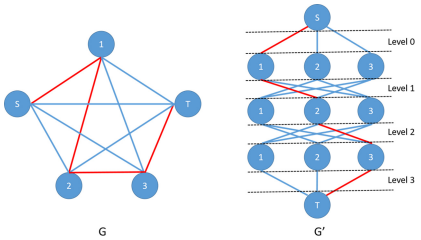

Fig. 2. Graph transformation.

2 Methodology

The main framework of our two-phase IntRoute method are shown in Fig. 1.

Phase 1: Contraction of Bus Stops. In the first phase of IntRoute, we contract multiple requests into a super node by using clustering approaches and determine a pickup bus stop for all the requests inside the super node. In each super node, passengers are asked to walk to the pickup bus stop and wait for buses. The walking time is counted into the total commuting time. We employ K-means clustering, as it consumes the least walking time compared with hierarchical clustering and density-peaks clustering. Then we exploit silhouette method and elbow method to determine the number of pickup stops, i.e., the cluster number n. Both methods indicate the best clustering number should be $n = 20$. Therefore, this phase finds 20 pickup stations that minimizes the passengers total walking time.

Phase 2: Design of One Alternate Route. In the real-world transportation network, an arbitrary node pair (i, j) may be connected (Fig. 2 left). To solve the problem via mathematical IP model, we introduce the *multi-level graph* G' (Fig. 2 right), where each level represents a possible sub-route from one stop to the next one. The red paths in G and G' are equal. The graph G' indicates the order of visiting the pick-up stops directly by the levels.

Modelling via Integer Programming. We denote the node set as $N = \{1, 2, \ldots, n\} \cup \{S, T\}$, where S and T represents the source and destination, respectively. The arc set is $A = \{(i, j) | i \in N,\ j \in N,\ \text{and}\ i \neq j\}$. The travel time from node i to node j is denoted as c_{ij}, and r_i is the number of passengers who want to go to the destination. $x_{ij}^l \in \{0, 1\}$ represents a binary variable, indicating whether the bus goes from node i to node j at level l on G'. $y_{ijk}^l \in \{0, 1\}$ represents a binary variable which denotes if request k travels through arc (i, j) at level l. $v_k^l \in \{0, 1\}$ represents a binary variable, indicating if request k is served at the level l. Formally, the IP model is formulated as:

$$\min \sum_l \sum_k \sum_{(i,j)\in A} y_{ijk}^l c_{ij} r_k, \quad s.t.$$

$$\textstyle\sum_{(i,j)\in A} x_{ij}^l = 1, for\ l = 0, 1, \ldots, n \tag{1}$$

$$\textstyle\sum_{(j,i)\in A} x_{ji}^l = \sum_{(i,k)\in A} x_{ik}^{l+1}, \forall i; \forall l \tag{2}$$

$$\sum_{(S,i)\in A} x_{Si}^{0} = \sum_{(i,j)\in A} x_{ij}^{1} \tag{3}$$

$$\sum_{(i,j)\in A} x_{ij}^{n-1} = \sum_{(j,T)\in A} x_{jT}^{n} \tag{4}$$

$$\sum_i \sum_l x_{ij}^{l} \leq 1, for \quad j = 1, 2, \ldots, n, T \tag{5}$$

$$v_k^l \leq v_k^{(l+1)}, \forall k, \forall l \tag{6}$$

$$\sum_k v_k^l = l, for\ l = 1, 2, \ldots, n \tag{7}$$

$$x_{ij}^{l} \leq v_i^l, for\ (i,j) \in A, l = 1, 2, \ldots, n \tag{8}$$

$$y_{ijk}^{l} \leq (x_{ij}^{l} + v_k^l)/2,\ \forall (i,j) \in A, \forall k, \forall l \tag{9}$$

$$y_{ijk}^{l} \geq (x_{ij}^{l} + v_k^l) - 1,\ \forall (i,j) \in A, \forall k, \forall l \tag{10}$$

$$x_{ij}^{l} \in \{0,1\};\ v_k^l \in \{0,1\};\ y_{ijk}^{l} \in \{0,1\} \tag{11}$$

where (1) ensures that each level is exactly passed once. (2)–(4) ensure that flow conservation of the graph. (5) ensures that each node can be entered at most once. (6) ensures that request k must be served at level $l+1$ if it is served at level l. (7) ensures that l requests are on the bus when the bus is serving level l. (8) ensures that if arc (i,j) is picked at level l, request i should be served at level l. (9) and (10) ensure that the request k is served at level l on the arc (i,j) if both x_{ij}^l and v_k^l are equal to one. (11) shows x_{ij}^l, v_k^l, and y_{ijk}^l are all binaries.

Optimization via Genetic Algorithm. We design a genetic algorithm (GA) to solve this IP problem. Specifically, the *chromosome* represents possible sequence of the node set $\{1, 2, \ldots, n\}$, and the *population* is a set of chromosomes.

As shown in Algorithm 1, a new chromosome can be generated by the *crossover* between two parent bus routes with a probability of crossover rate R_c. The *mutation* is defined as the position exchange between two randomly selected near-by bus stoops with a probability of R_m. There are two steps in our GA: (1) the randomly population initialization with a given size M, and (2) the population evolution for T generations by crossover and mutation according to a fitness function. We adopt a 2-OPT technique to avoid the cross sub-paths. Besides, we propose a decomposition technique that clusters the optimal route into three sub-route via k-means. We concatenate the clusters in a reverse order, i.e., from the destination node to the start node. This strategy greatly reduces the travel time.

Algorithm 1: Genetic Algorithm for Solving the IP Problem

Input: $T = 1000,\ R_c = 0.1,\ R_m = 0.05$
Output: The chromosome with the best fitness function

```
Initialize the population with size M = 500;
for i = 1 to T do
    Select new population P_i from P_{i-1};
    for individual p ∈ P_i do
        offspring ⟵ Crossover(p, R_c);
        offspring ⟵ Mutate(offspring, R_m);
        p ⟵ offspring;
    end
end
return The best chromosome.
```

Table 1. Routing time before optimization

Bus route	C'	C
2153142, 2155252, 2148445, 2153226, 2121125, 2074117, 212225, 211220, 211118, 2137134, 212746, 2150145, 2145561, 2150302, 2190145, 2135206, 213447, 203833, 200721, 201635, CBD	33.62 min	31.53 min

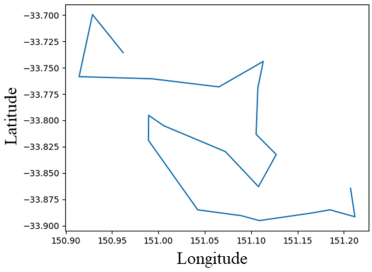

Fig. 3. Bus route without transfer

Table 2. Commute time

Methods	Time
Original route	31.53
Greedy algorithm [1]	53.41
GA	31.27
GA + 2-OPT	33.62
GA + decomposition	18.06

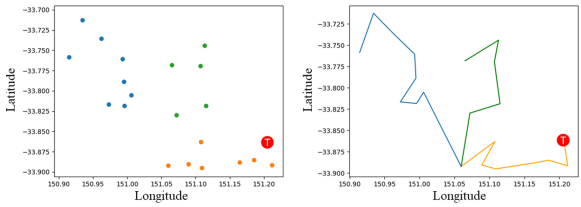

Fig. 4. Bus routes after optimization

3 Experiments and Analysis

Data. The experiment is performed based on publicly available real-world commuting data, retrieved from the card-based transit payment system[1] in Sydney, Australia, including approximately three million trips.

Results. The routing time before optimization are listed in Table 1, where routes are represented by the IDs of the bus stops. Here C denotes the time cost without transfer and without optimization (also demoed in Fig. 3), and C' denotes that of the original commute with transfers. The new route after our optimization is shown in Fig. 4, while the time costs of the bus routes optimized by different methods are listed in Table 2.

Conclusions. Our IntRoute method greatly reduces the time expense for passengers from 31.53 min to 18.06 min on average, saving about 43% of commute time. In future, we plan to investigate more optimization methods for further improving our solutions.

References

1. Li, L., Fu, Z.: The school bus routing problem: a case study. J. Oper. Res. Soc. **53**(5), 552–558 (2002). https://doi.org/10.1057/palgrave.jors.2601341
2. Liu, J., Mao, J., Du, Y.T., Zhao, L., Zhang, Z.: Dynamic bus route adjustment based on hot bus stop pair extraction. In: Li, G., Yang, J., Gama, J., Natwichai, J., Tong, Y. (eds.) DASFAA 2019. LNCS, vol. 11448, pp. 562–566. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18590-9_87
3. Liu, W., Chawla, S.: A quadratic mean based supervised learning model for managing data skewness. In: Proceedings of SIAM SDM Conference (2011)

[1] https://opendata.transport.nsw.gov.au/dataset/opal-tap-on-and-tap-off.