Multi–criteria Route Planning in Bus Network

Vo Khoa, Tran Pham, Huynh Nguyen, Tran Hoai

Multi–criteria route planning in bus network

Vo Dang Khoa*, Tran Vu Pham, Huynh Tuong Nguyen, and Tran Van Hoai

Faculty of Computer Science & Engineering - Ho Chi Minh City University of Technology
268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Vietnam
{khoa.v.dang}@gmail.com
{t.v.pham,htnguyen,hoai}@cse.hcmut.edu.vn
http://www.cse.hcmut.edu.vn/

Abstract. In this paper, we consider the problem of finding itineraries in bus networks under multiple independent optimization criteria, namely arrival time at destination and number of transfers. It is also allowed to walk from one stop to another if the two stops are located within a small distance. A time–dependent model is proposed to solve this problem. While focusing on the network where the size of the Pareto set in the multi–criteria shortest path problem might grow exponentially, we develop an efficient algorithm with its speed–up techniques. An evaluation on the qualities of found paths and the empirical results of different implementations are given. The results show that the allowance of walking shortcuts between nearby stops gives a better route planning.

Key words: Time–dependent model, shortest path problem, public transport system, bus system, labelling algorithm

1 Introduction

Route planning becomes essential for bus users when the size of the timetable gets larger. A manual planning is mainly based on individual experience and is hardly optimal. Therefore, the planning should be done automatically and in an algorithmic way. The user then makes a query that consists of a *source bus stop* (or stop) A, a *destination stop* B and a *departure time* t_A at A. The system answers with the optimal route planning to reach B under the optimization of multi–criteria, namely arrival time at B and number of transfers. Tradeoffs of the multi–criteria optimization is that the improvement of one criterion comes at the expense of decreasing other criteria. For instance, the planning with an earlier arrival time at destination might have a greater number of transfers, whereas the one with a smallest number of transfers might have a later arrival time at destination. A set of Pareto–optimal paths [11] then allows to focus on important tradeoffs without considering the full range of all possible values of criteria. The multi–criteria route planning problem in the bus network then can be solved by transforming the problem to a multi–criteria shortest path problem in a graph model.

The multi–criteria shortest path problem is an extension of the classical shortest path problem. The problem aims to find a set of *non–dominated* (Pareto–optimal) paths (Pareto set) from a *source vertex* to a *destination vertex* under *multiple* criteria. A path p dominates a path q iff p's weight is less than or equal q's weight in all criteria and one of the inequalities is strict. A path is called Pareto–optimal if it is not dominated by any other paths. The standard way to find the Pareto set in a non–negative weighted graph is *labelling algorithm*. The algorithm keeps candidate paths which can be further expanded in a priority queue of labels. Labelling algorithm is distinguished by the order paths are extracted from the queue and the policy for the expansion at each vertex. In particular, *label setting method* [10, 11, 14, 15] chooses paths in a *lexicographical* order. Given two path p and q with p's weight vector (x_p, y_p) and q's weight vector (x_q, y_q) respectively, p is lexicographically less than q iff $x_p < x_q$ or $(x_p = x_q$ and $y_p \leq y_q)$. Lexicographical ordering assures that established paths are the Pareto–optimal paths [11]. On the other hand, *label correcting method* [9, 11–13] extracts paths in the FIFO order or the minimum weighted sum aggregate order, the chosen paths might be dominated by other paths later. As a result, the established paths which contain the chosen path have to be corrected. Regarding the expansion policy, it is either *label–selection* [10, 11, 14] or *vertex–selection* [9, 12]. In label–selection policy, labels belonging to a particular vertex is treated separately. That means there is only one path expanded at each time for each outgoing arc. In contrast, vertex–selection policy extracts all paths which have a same destination vertex from the queue at the same time and then expands all of them for each outgoing arc.

As for the graph model, there are two main approaches: *time-expanded* [2, 3], and *time-dependent* [4–6]. Hannemann and Schulz [1] gave a great survey on these models. In time–expanded approach, a static weighted graph is used to model *time events* (departure/arrival pairs) in the timetable. Schulz et al. [2] used the model to solve the simplified version of railway system problem. Hannemann et al. [3] then extended the simplified model with the involvement of ticket cost, and number of transfers as well as traffic days. In time–dependent approach, weights of arcs in the arc set E are determined by function $f : E \times T \rightarrow T$, with time set T in the timetable. Time–dependent model was first proposed by Brodal and Jacob [4]. The aim of their study is to propose a more effective model in comparison with the time–expanded model in [2]. Brodal and Jacob argued that in the simplified case where transfer from a train to another is not specified, Dijkstra's algorithm [16] considers many redundant arcs in the time–expanded graph. Pyrga et al. [5] then proposed an extended model which allows transfers between trains, and then they improved their model with the involvement of traffic days in [8]. Disser et al. [6] extend the model in [8] by introducing the criterion of *"reliability of transfers"* which presents the probability of catching all trains of the planing.

In this paper, we formulate a multi–criteria route planning problem in a bus network as a multi–criteria shortest path problem. A time–dependent model and a labelling algorithm with its speed–up techniques are then proposed to solve the

problem. Our proposed model allows the possible transfer between nearby stops that are within a walking distance. Empirical results are therefore evaluated and presented using Ho Chi Minh City (HCMC) bus network. From the obtained results, we also discuss about the quality of the route planning when users are able to walk between stops. The remaining of the paper is organised as follows. In Section 2, we present how the bus network is modelled. The algorithm and speed–up techniques then are proposed in Section 3. The experimental studies are given in Section 4. Finally, the conclusion is presented in Section 5.

2 Route planning problem

2.1 Bus system

Generally, a bus system is formulated by $\mathfrak{I} = (N, R, T, TT)$ in which

- $N = (S, C)$ is a directed graph that presents the bus stop network where the vertex set S is a set of stops, and the arc set $C \subseteq S^2$ is a set of stop connections.
- R is a set of routes. For each route $x \in R$, let S_x and $C_x \subseteq S_x^2$ be a set of stops and a set of connections of route x such that $S = \bigcup_{x \in R} S_x$ and $C = \bigcup_{x \in R} C_x$. A route $x \in R$ is defined by an acyclic path $x = \langle s_1, s_2, ..., s_k \rangle$ on graph N with $(s_i, s_{i+1}) \in C_x, \forall i \in \{1, ..., k-1\}$. Route x means there are a set of buses starting their route from the stop s_1, then visiting consecutively $s_2, ..., s_{k-1}$ and finishing at s_k.
- $T \subset \mathbb{Z}^+$ is a set of time points appearing in the timetable. T depends on the period the timetable is considered, daily or weekly.
- TT is the timetable. The timetable TT consists of *elements* c which has the form $c = (Z, S_d, S_a, t_d, t_a)$. An element c then means there is a bus running on route $Z(c)$ which departs from stop $S_d(c)$ at time $t_d(c)$, and then arrives at stop $S_a(c)$ at time $t_a(c)$ in the timetable TT. For each $c \in TT$, c has the following constraints: $Z(c) \in R$; $S_d(c)$, $S_a(c) \in S_{Z(c)}$; $t_d(c), t_a(c) \in T$; $t_a(c) \geq t_d(c)$.

2.2 Time–dependent model

The problem based on time–expanded model is simpler to model [2], but needs a higher space consumption due to the graph model structure [4]. Empirical results also show that time–dependent model gives a better performance in running time [7, 8]. In this paper, we propose a time–dependent model which is extended from the model in [5], but allows walking between nearby stops. The idea of the extension is that arcs which model the walking shortcuts are introduced. Let $G = (V, E)$ be the directed graph that models the bus system $\mathfrak{I}$, and let $\xi : V \to S$ define the injective function mapping a vertex $u \in V$ to a stop $A \in S$. The graph G is constructed by the following steps:

1. Let $G_R = (V_R, E_R)$ be the directed graph that models the routes in R, with $V_R = \bigcup_{x \in R} W_x$ and $E_R = \bigcup_{x \in R} F_x$. For each route $x \in R$, create a *route vertex* $r_A^x \in W_x$ for every stop $A \in S_x$, and let $\xi(r_A^x) = A$, then create a *route arc* $(r_A^x, r_B^x) \in F_x$ for every connection $(A, B) \in C_x$.
2. Let $G_S = (V_S, E_{walk})$ be the directed graph that models the stops in S. Create a *stop vertex* $s_A \in V_S$ for every stop $A \in S$, and let $\xi(s_A) = A$, then create a *walking arc* $(s_A, s_B) \in E_{walk}$ for every pair of stop $A, B \in S$ that is allowed to walk between.
3. The graph $G = (V, E)$ is given by $G = G_R \oplus G_S$ with $V = V_R \bigcup V_S$, and $E = E_R \bigcup E_{walk} \bigcup E_{in} \bigcup E_{off}$ where $E_{in} \subseteq V_S \times V_R$ is the set of *get–in* arcs, and $E_{off} \subseteq V_R \times V_S$ is the set of *get–off* arcs. Then, create an arc $(u, v) \in E_{in}$ and an arc $(v, u) \in E_{off}$ for every pair of vertices $u \in V_S$ and $v \in V_R$ if $\xi(u) = \xi(v)$.

Figure 1 illustrates the graph model of the connection from stop A to stop B in which two routes α, β visit A then B; route γ visits A but not B; route δ visits B but not A; and walking is allowed from A to B and from B to A.

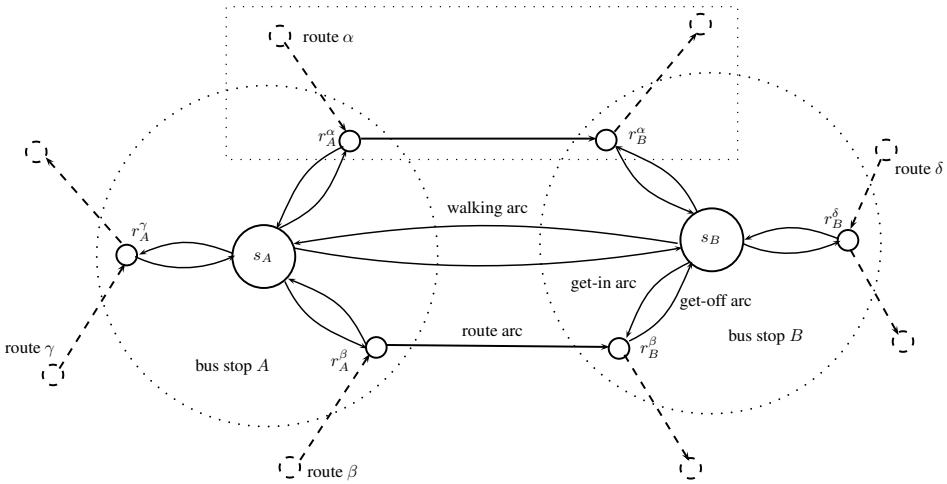

Fig. 1. The model for the connection from stop A to stop B.

The graph model has the following properties: $\bigcap_{x \in R} W_x \bigcap V_S = \emptyset$, and $\bigcap_{x \in R} F_x \bigcap E_{walk} \bigcap E_{in} \bigcap E_{off} = \emptyset$. With these properties, the vertices and the arcs of the graph G capture all possible states and possible actions of a bus user respectively. The semantics of the vertices and the arcs are given as follows:

- A stop vertex $u \in V_S$ presents the state that one is waiting at stop $\xi(u)$.
- A route vertex $u \in W_x$ presents the state that one visits stop $\xi(u)$ on a bus of route x.
- A get–in arc $(u, v) \in E_{in}$, with $u \in V_S$ and $v \in W_x$, presents the action that one gets in a bus of route x at stop $\xi(u)$.

– A get–off arc $(u, v) \in E_{off}$, with $u \in W_x$ and $v \in V_S$, represents the action that one gets off a bus of route x at stop $\xi(v)$.
– A walking arc $(u, v) \in E_{walk}$ represents the action that one walks from stop $\xi(u)$ to stop $\xi(v)$.
– A route arc $(u, v) \in E_R$, with $u, v \in W_x$, represents the action that one travels from stop $\xi(u)$ to stop $\xi(v)$ on route x.

Given a path $p = \langle v_1, v_2, ..., v_k \rangle$, $k > 2$, on the graph model G, path p gives a route planning for a user query from stop A to stop B with the departure time t_A at A with the following constraints: $v_1, v_k \in V_S$; $\xi(v_1) = A$; $\xi(v_k) = B$; $(v_i, v_{i+1}) \in E, \forall i \in \{1, ..., k-1\}$.

2.3 Multi–criteria shortest path problem

Travel time – We consider three kinds of travel time, namely *routing time*, *transfer time* and *walking time*, as below.

bus	Z	S_d	S_a	t_d	t_a
...	...	...	...	...	...
bus 1	α	A	B	8:05	8:30
bus 2	β	A	B	8:18	8:39
bus 3	α	A	B	8:20	8:55
...	...	...	...	...	...

Table 1. A part of the timetable TT which relates to the connection from stop A to stop B.

bus	t_d	t_a
...	...	...
bus 1	8:05	8:30
bus 3	8:20	8:55
...	...	...

Table 2. The time table $TT^{\alpha}_{A,B}$ of route α from stop A to stop B.

– Let $\theta^{\alpha}_{A,B}(t_A)$ be the routing time from stop A to stop B on route α with the arrival time t_A at A, then

$$\theta^{\alpha}_{A,B}(t_A) = min\{t' \mid (t, t') \in TT^{\alpha}_{A,B}, t_A \leq t\} - t_A \quad (1)$$

$$TT^{\alpha}_{A,B} = \{(t_d(c), t_a(c)) \mid c \in TT, S_d(c) = A, S_a(c) = B, Z(c) = \alpha\} \quad (2)$$

where $TT^{\alpha}_{A,B}$ is the timetable of route α from A to B. If $TT^{\alpha}_{A,B} = \emptyset$, that means there is no connection from A to B via route α in the timetable. In other words, one cannot catch any bus of route α to travel from A to B. The *min* operation in Formula 1 yields the pair (t, t') in $TT^{\alpha}_{A,B}$ such that its departure time t is after t_A and its arrival time t' is minimum. Let the minimum arrival time (if exists) be t^*, the time to travel from A to B is $t^* - t$ and the waiting time at A before catching a bus of route α to B is $t - t_A$, then

$$\theta^{\alpha}_{A,B}(t_A) = t^* - t + t - t_A = t^* - t_A.$$

Table 1 gives a part of the timetable TT which relates to the connection from stop A to stop B. The timetable $TT^{\alpha}_{A,B}$ is presented in Table 2. For the route α, suppose we arrive at A at time t_A = 8:15, at the time t_A we cannot catch bus 1, since it has already left A at 8:05 and the next bus of route α is bus 3. The routing time from A to B on route α when we arrive at A at 8:15 then is 8:55 - 8:15 = 40 minutes.

- At every stop A, it is only possible to get in a bus of a route α if we arrive at A earlier than the departure time of that bus by a given non–negative transfer time, denoted by τ_A.
- Let $\omega_{A,B}$ be the time to walk from stop A to stop B, the walking time is computed by the walking distance from A to B divided by the average walking speed.

Let $f : E \times T \to T$ be the function that determines the travel time weight of an arc $(u, v) \in E$ at time $t_u \in T$ where t_u is the time arc (u, v) is traversed, then

$$f(u, v, t_u) = \begin{cases} 0 & \text{if } (u, v) \in E_{off} \\ \tau_{\xi(u)} & \text{if } (u, v) \in E_{in} \\ \omega_{\xi(u),\xi(v)} & \text{if } (u, v) \in E_{walk} \\ \theta^x_{\xi(u),\xi(v)}(t_u) & \text{if } (u, v) \in F_x \subseteq E_R \end{cases} \tag{3}$$

Number of transfers – Let $g : E \to \{0, 1\}$ be the function that determines the number of transfer weight of an arc $(u, v) \in E$, then:

$$g(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E_{in} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Combine two criteria – Let vector function $w : E \times T \to T \times \{0, 1\}$ be weight function of an arc $(u, v) \in E$ at time $t_u \in T$ where w^1 is travel time weight and w^2 is the number of transfers weight, the function w is given by:

$$w(u, v, t_u) = (f(u, v, t_u), g(u, v)) \tag{5}$$

Multi–criteria shortest path problem – Given a path $p = \langle v_1, v_2, ..., v_k \rangle$, $k > 2$, from vertex $v_1 = s$ to vertex $v_k = d$ with $w^{arrival}(p, t_s)$ being the arrival time at d if the departure time at s is t_s, and $w^{transfer}(p)$ being the number of transfers on p. The weight vector $w(p, t_s)$ of path p with the departure time t_s at s is formulated by

$$\begin{aligned} w(p, t_s) &= (w^{arrival}(p, t_s), w^{transfer}(p)) \\ &= (t_s, 0) + \sum_{i=1}^{k-1} w(v_i, v_{i+1}, t_{v_i}) \\ &= (t_s, 0) + \sum_{i=1}^{k-1} (f(v_i, v_{i+1}, t_{v_i}), g(v_i, v_{i+1})) \end{aligned} \tag{6}$$

where $f(v_i, v_{i+1}, t_{v_i})$, $g(v_i, v_{i+1})$ are given by Formula 3 and Formula 4 respectively, and

$$t_{v_{i+1}} = \begin{cases} t_s & \text{if } i = 0 \\ t_{v_i} + f(v_i, v_{i+1}, t_{v_i}) & \text{otherwise} \end{cases} \tag{7}$$

with t_{v_i} being the arrival time at vertex v_i, or the time the arc (v_i, v_{i+1}) is traversed. Then, the arrival time t_{v_i} is used as the input to compute the arrival time $t_{v_{i+1}}$ at vertex v_{i+1}, and so on.

Let $P_{s,d}$ be a set of all possible paths from vertex s to vertex d. The set of Pareto–optimal paths $\Pi_{s,d} \subseteq P_{s,d}$ is formulated by:

$$\Pi_{s,d} = \{p \in P_{s,d} \mid \nexists q \in P_{s,d} : q \prec p\} \tag{8}$$

where $\prec$ is a *strictly dominance* relation. Path q strictly dominates path p, denoted by $q \prec p$, or p is dominated by q iff:

$$\begin{gathered} w^{arrival}(q, t_s) \leq w^{arrival}(p, t_s) \\ w^{transfer}(q) \leq w^{transfer}(p) \\ w(q, t_s) \neq w(p, t_s) \end{gathered}$$

In order to avoid arcs with negative travel time weights in the graph model, Assumption 1 on the elements in the timetable TT must be held [5]. It implies that buses of the same route need to follow the FIFO property in the traffic flow.

Assumption 1 *Given two elements $c_1, c_2 \in TT$ with $Z(c_1) = Z(c_2), S_d(c_1) = S_d(c_2)$ and $S_a(c_1) = S_a(c_2)$. If $t_d(c_1) \leq t_d(c_2) \Rightarrow t_a(c_1) \leq t_a(c_2)$ [5].*

Given a path $p \in \Pi_{s,d}$, p follows the optimality principle [11] if $w^1(u, v, t_u) \geq 0$ and $w^2(u, v) \geq 0$ for every arc $(u, v) \in p$. Indeed, with Assumption 1, the travel time function f is non–negative. Also, the function g is non–negative constant. The multi–criteria shortest path problem then can be solved by labelling algorithm.

3 Algorithm

3.1 Labelling algorithm

For the labelling algorithm, we use the label setting method and the label selection policy. The used notations are defined as below.

- $\Pi_{s,i}$: a set of non–dominated paths from the source s to a vertex i.
- $\Diamond$: relation that concatenates a path p with an arc (i, j) or another path q.
- Q: the queue of candidate paths that can be further expanded.
- $\prec$: the dominance relation between two paths.

The pseudo code of the proposed algorithm is given in Algorithm 1. The key idea of the algorithm is one vertex associated with several labels, each corresponding with a non–dominated path from the source to that vertex. Since the number of non–dominated paths in $\Pi_{s,d}$ is nondeterministic, the algorithm only terminates when Q is empty (Line 4). For each iteration, path $p = \langle s, ..., i \rangle$ is extended along all its outgoing arcs (i, j) (Line 8). To relax an arc (i, j), we check if the new extended path $q = p \Diamond (i, j)$ is not dominated by any path $u \in \Pi_{s,j}$ (Line 12). $\Pi_{s,d} \neq \emptyset$ means some non–dominated paths have been found. For

a given path q expanded via an arbitrary arc (j,k), if q is dominated by any path in $\Pi_{s,d}$, the path $q\Diamond(j,k)$ is also dominated since the weight of arc (j,k) is non–negative, and so on. Therefore, we can use paths in $\Pi_{s,d}$ as the upper bound to early prune all paths if they are dominated by any path in $\Pi_{s,d}$, since q will lead to dominated paths which have form of $q\Diamond\langle j,...,d\rangle$ in $\Pi_{s,d}$. This early prune is illustrated in Line 10. If a path q satisfies two conditions in Line 10 and Line 12, that means q might lead to a non–dominated path at $\Pi_{s,d}$. Therefore q is put into Q for further expansion (Line 14). There is still the case that some paths in $\Pi_{s,j}$ are dominated by q, Line 13 is supposed to remove all these paths and update $\Pi_{s,j}$.

Algorithm 1: The labelling algorithm

Input : a graph G with a source s and a destination d and a departure time t_s.
Output: the Pareto set of optimal paths $\Pi_{s,d}$.

```
1  Π_{s,s} ← {⟨s⟩};
2  Π_{s,i} ← ∅, ∀i ∈ V \ {s};
3  Q ← {⟨s⟩};
4  while Q ≠ ∅ do
5      p ← the lexicographically lowest path in Q;
6      Q ← Q \ {p};
7      i ← the destination vertex of p;
8      for (i,j) ∈ E do
9          q ← p◇(i,j);
10         if ∄u ∈ Π_{s,d} : u ≺ q then
11             continue;
12         if ∄u ∈ Π_{s,j} : u ≺ q then
13             Π_{s,j} ← (Π_{s,j} ∪ {q}) \ { u ∈ Π_{s,j} | q ≺ u};
14             Q ← Q ∪ {q};
```

3.2 Speed–up techniques

1. **Backward get–off arc avoidance**: In the graph model, get–off arcs have zero weights. The labelling algorithm might explore back to stop vertices via these arcs. These labels are absolutely dominated later. The search therefore should avoid these arcs.
2. **Upper bounds for criteria**: The realistic assumptions can be used to set constraints on criteria. For example, it is undesirable for a path which has the number of bus transfers greater than 4 times or the travel time more than 3 hours. Given an upper bound for travel time α and an upper bound for number of transfers β, then any path p with $w^{arrival}(p,t) > \alpha + t$ or $w^{transfer}(p) > \beta$ should be early pruned.
3. **Search area reduction**: Depending on the locations of source and destination, only a part of the bus network is used for searching. In this paper, we propose the search area as a rectangle. The idea is as follows. Suppose we has a query for the route planning from stop A to stop B, then we can calculate the differences in x and y–coordinate of A and B, denoted as dx and dy respectively. Given $0 < \epsilon < 1$, the target search area is then obtained

via the scaling up of the x and y–coordinate based on ϵ and the ratio dx/dy as illustrated in Figure 2.

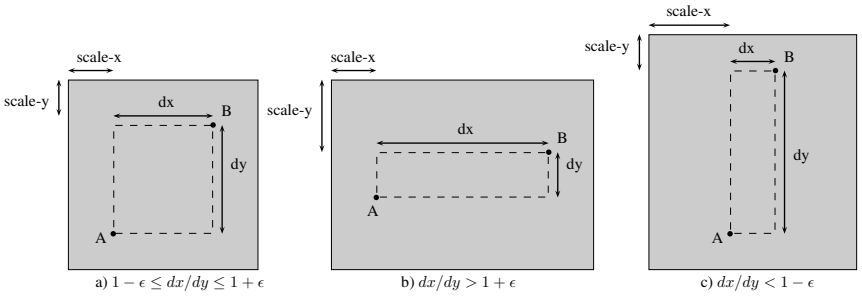

Fig. 2. Three possible cases in which the target search area is scaled up based on the ratio dx/dy.

4 Computational study

4.1 The graph instance and the testing data set

HCMC bus network consists of 4,090 stops, 220 routes and 8,833 stop connections. It is allowed for walking shortcuts between stops in the radius of 150 metres. Regrading the daily timetable, at some stops like the begin and the end of a route, the departure and the arrival times are fixed. These are used to estimate the departures and the arrival times of the remaining stops in the route. The graph model has 13,140 vertices and 29,161 arcs. Assume that the constructed graph and the timetable are stored in the primary memory and binary heap is used to implement the queue of candidate paths. We consider three criteria of measurements, namely the number of created labels (vertex visits), the number of queue operations and the running time. We also evaluate the quality of the solution, namely arrival time and number of transfers of found paths. The data set for testing is made up from 1,000 random user requests where the source, the destination and the query time are generated randomly. The empirical results then are the average value of the data set.

4.2 Empirical results

The empirical study is conducted on 2.6 GHz dual–core Intel Core i5 4GB RAM on MAC OS X under Java Runtime Environment 1.6 (JRE 1.6). The tested database is MySQL 5.2.

First we compare the proposed model with Pyrga's model [5] in which it is not allowed to walk between nearby stops. In this case we use the implementation of label setting without any speed–up techniques. As seen from Table 3, the proposed model is inferior to Pyrga's model in term of the performance because it has additional walking arcs. But the proposed one produces more efficient route planning with lesser travel time and smaller number of transfers. Besides,

there are 1,322 paths which contains walking arc amongst 1,376 non–dominated paths. That means the addition of walking options produce a more effective route planning. However, the appearances of walking arcs in majority of optimal paths also indicate that the allocations of stops in the network is not well designed.

model	created label	queue operations	running time in ms	travel time in minutes	number of transfers
Pyrga's model	30,034	14,084	55.930	74	2.686
proposed model	42,936	15,470	75.882	49	1.450

Table 3. The comparison between the Pyrga's model and the proposed model.

Since there are only 1,376 non–dominated paths over 1,000 queries. In other words, only 1.357 non–dominated paths in the Pareto set per query on average. We compare the average quality of found paths in the Pareto set with the average quality of paths under single optimization. For the single optimization, we use Epsilon–constraint method in which arrival time is optimized and number of transfers is bounded. In this experiment, the used bound value is 4. As shown in Table 4, the average number of transfers of paths in the Pareto set is less than that of the paths which only optimize travel time. That means the Pareto set contains desirable paths with smaller number of transfers which are left out if we only consider the problem in single–criteria manner.

algorithm	created label	queue operations	running time in ms	travel time in minutes	number of transfers
single–criteria	29,757	11,196	47.056	47	2.920
multi–criteria	42,936	15,470	75.882	49	1.450

Table 4. The comparison between multi–criteria and single–criteria.

variant	backward avoidance	criteria upper bound	search area	created label	queue operations	running time in ms
plain version	no	no	no	42,936	15,470	75.882
backward avoidance	yes	no	no	36,249	15,470	75.417
upper bound	no	$\alpha = 5$, $\beta = 3$ *hours*	no	42,783	15,397	75.632
search area reduction	no	no	yes	31,057	11,138	51.822
optimized version	yes	$\alpha = 5$, $\beta = 3$ *hours*	yes	26,136	11,081	51.559

Table 5. The comparison between different speed–up techniques.

Table 5 gives the comparison between the different speed–up techniques with various parameters. The results show that the backward avoidance reduces significantly the number of created labels, but does not reduces the number of queue operations. This is consistent with what discussed in Section 3.2 because all backward get–in arcs will lead to dominated paths which are discarded before being put into the queue. Using the transfer bound and travel time bound decreases both the number of created labels and queue operations, that leads to improve the running time, but these improvements are not so significant. The search area reduction brings a huge speed–up. The drawback of the technique is

losing optimal paths if the searching area is not estimated properly. Therefore, in the empirical result, we adjust the target search area in the way that no solution is lost. The technique reduces 25% number of created labels, 30% number of queue operations and 30% running time in comparison to the plain version.

5 Conclusion

We have developed in this paper the model for transforming a route planning problem in a bus network to a multi–criteria shortest path problem. An modified time–dependent model and an efficient labelling algorithm are proposed and evaluated. The results show that the size of Pareto set in HCMC bus network is quite small, and the addition of walking shortcuts gives a much better route planning. The quality of paths produced by the algorithm lies in the correctness of the timetable. However, in reality, arrival and departure times of buses might disobey the timetable when traffic condition is involved. Thus, in future work, we will develop the model combining the timetable and real–time data from the bus tracking system. Besides, the appearance of walking shortcuts in the majority of route planning also raises the question whether the allocation of stops is efficient.

Acknowledgments

The authors gratefully acknowledge the helpful comments offered by Professor Hai Le Vu in Intelligent Transport Systems Lab of Swinburne University of Technology, Melbourne, Australia. This research is supported by Ho Chi Minh City University of Technology, under grant number T-KHMT-2014-28 and T-KHMT-2014-32.

References

1. M.M. Hannemann, F. Schulz, D. Wagner, and C. Zaroliagis. *Timetable Information: Models and Algorithms. Algorithmic Methods for Railway Optimization.* In Lecture Notes in Computer Science, vol. 4359, pp. 67–90, 2007.
2. F. Schulz, D. Wagner, and K. Weihe. *Dijkstra's Algorithm On–Line: An Empirical Case Study from Public Railroad Transport.* In Algorithm Engineering Lecture Notes in Computer Science, vol. 1668, pp. 110–123, 1999.
3. M.M. Hannemann, and M. Schnee. *Finding All Attractive Train Connections by Multi-criteria Pareto Search.* In Lecture Notes in Computer Science, vol. 4359, pp. 246–263, 2007.
4. G.S. Brodal, and R. Jacob. *Time–dependent networks as models to achieve fast exact time-table queries.* In Electronic Notes in Theoretical Computer Science, vol. 92, pp. 3–15, 2004.
5. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. *Towards Realistic Modeling of Timetable Information through the Time–Dependent Approach.* In Electronic Notes in Theoretical Computer Science, vol. 92, pp. 85–103, 2004.

6. Y. Disser, M.M. Hannemann, and M. Schnee. *Multi–criteria Shortest Paths in Time–Dependent Train Networks. Experimental Algorithms.* In Lecture Notes in Computer Science, vol. 5038, pp. 347–361, 2008.
7. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. *Experimental comparison of shortest path approaches for timetable information.* In 6th Workshop on Algorithm Engineering and Experiments (ALENEX04), SIAM, pp. 88–99, 2004.
8. E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. *Efficient Models for Timetable Information in Public Transportation Systems.* In Journal of Experimental Algorithmics (JEA), vol. 12, Article No. 2.4, 2008.
9. A.J.V. Skriver and K.A. Andersen. *A label correcting approach for solving bicriterion shortest path problems.* In Computers & Operations Research, vol. 27, pp. 507–524, 2000.
10. X. Gandibleux, F. Beugnies, and S. Randriamasy. *Martins' algorithm revisited for multi–objective shortest path problems with a MaxMin cost function.* In 4OR, vol. 4, pp. 47–59, 2006.
11. E.Q.V. Martins and J.L. Santos. *The labeling algorithm for multicriteria shortest path problem.* Departamento de Matematica, Universidade de Coimbra, Portugal, 1999.
12. J.Brumbaugh and Smith. *An empirical investigation of some bicriterion shortest path algorithms.* In European Journal of Operational Research, vol. 43, pp. 216–224, 1989.
13. J. Mote, I. Murthy, and D.L. Olson. *A parametric approach to solving bicriterion shortest path problems.* In European Journal of Operational Research, vol. 53, pp. 81–92, 1991.
14. J.M. Paixao and J.L. Santos. *Labeling methods for the general case of the multiobjective shortest path problem–a computational study.* In Intelligent Systems, Control and Automation: Science and Engineering, vol. 61, pp. 489–502, 2013.
15. G. Mali, P. Michail and C. Zaroliagis. *Faster multiobjective heuristic search in road maps.* In Proceedings ICT, vol. 3, pp. 67–72, 2012.
16. E.W. Dijkstra. *A note on two problems in connection with graphs.* In Numerische Mathematik, vol. 1, pp. 269–271, 1959.