

Timetable-based Transit Assignment Using Branch & Bound

Authors:

Markus Friedrich, PTV AG

Stumpfstr. 1, D-76131 Karlsruhe, Germany

Phone ++49-721-9651316, fax ++49-721-9651299, email markus.friedrich@ptv.de

Ingmar Hofsäß, PTV AG

Stumpfstr. 1, D-76131 Karlsruhe, Germany

Phone ++49-721-9651312, fax ++49-721-9651299, email ingmar.hofsaess@ptv.de

Steffen Weckeck, PTV AG

Stumpfstr. 1, D-76131 Karlsruhe, Germany

Phone ++49-721-9651339, fax ++49-721-9651299, email steffen.weckeck@ptv.de

Abstract:

Transit assignment procedures need to reflect the constraints imposed by line routes and timetables. They require specific search algorithms considering transfers between transit lines with their precise transfer time. The paper will present such an assignment procedure for transit networks using a timetable-based search algorithm. In contrast to existing timetable-based search methods employing a shortest path algorithm, the described procedure constructs connections using branch & bound techniques. This approach significantly reduces computing time, thus facilitating the use of timetable-based assignment for big networks. At the same time it produces better results in cases where slow but cheap or direct connections compete with fast connections which are more expensive or require transfers.

Keywords:

Transit Assignment, Timetable-based Assignment, Public Transport Modeling

1 INTRODUCTION

Assignment procedures form the core of any comprehensive transportation model. Modelling a person's travel behaviour on their journey from an origin to a destination, such procedures allow to distribute a given travel demand to a network. This results not only in traffic volumes for road links and transit lines but also in indicators describing the service quality of a network.

For private transport assignment, often denoted as highway assignment, a huge number of publications (e.g. [1], [2]) can be found describing various solutions ranging from simple incremental procedures to equilibrium approaches and dynamic traffic assignment. Publications on public transport assignment, however, are comparatively rare. After several chapters on private transport assignment, textbooks often claim that the transit assignment problem is similar to the auto assignment problem. This, of course, is only partly true as a comparison of travel behaviour between car drivers and transit passengers shows: Car drivers may depart at any time and are free to choose a route which appears convenient to them. Travel behaviour of transit passengers, on the other hand, is strongly restricted by the line network and the timetable. In addition to choosing a route, transit passengers also select a departure time from the timetable, that is, they pick a connection. While a route only describes the spatial course of a trip within a network, a connection additionally encompasses temporal constraints such as departure and arrival times at the origin stop, the transfer stops, and the destination stop.

Assignment procedures for a transit network need to reflect the constraints imposed by line routes and timetables. They require specific search algorithms considering transfers between transit lines with their precise transfer time. This type of *timetable-based* search algorithm can be found in passenger information systems provided by an increasing number of transit operators (e.g. www.bahn.de, www.sbb.ch, www.ns.nl). Here, the search algorithms are applied and optimised for a one-to-one type of problem, i.e. they calculate connections for one origin-destination (o-d) pair. Assignment in transport planning, however, needs to examine an entire network and its connections between all traffic zones. This extends the search problem to a many-to-many problem, which can produce long computing times for big nationwide networks. To ensure acceptable computing times, assignment models often apply simplified search algorithms based on *average values* for the transfer time. These approaches reduce the problem by assuming that the wait times at the boarding stop or at transfer stops depend on the headway of the following transit line. This assumption speeds up computation but fails to consider the co-ordination of the timetable, which is essential in transit networks with long headways.

This paper will present an assignment procedure for transit networks using a timetable-based search algorithm. In contrast to existing timetable-based search methods employing a shortest path algorithm, the described procedure constructs connections using branch & bound techniques. This approach significantly reduces computing time, thus facilitating the use of timetable-based assignment for big networks. At the

same time it produces better results in cases where slow but cheap or direct connections compete with fast connections which are more expensive or require transfers.

2 EXISTING APPROACHES

Existing approaches for transit assignment can be divided into the categories line-based and timetable-based:

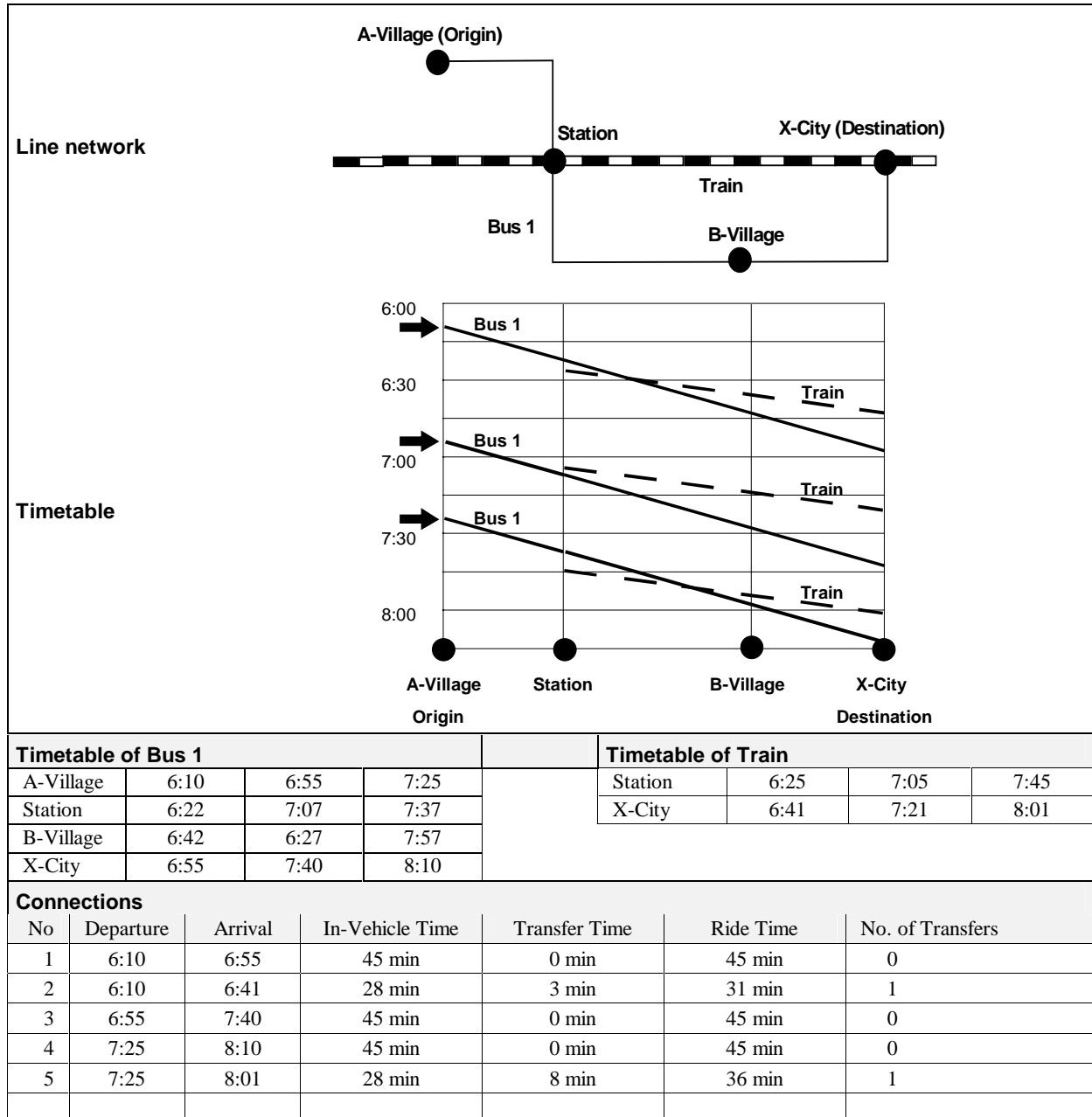
1. *Line-based* assignment simplifies the search by using *average values* for the transfer time (see [3], [4]). It models each transit line through a sequence of stops, through the running times between the stops and through the headway of the transit line. Transit lines with no fixed-rhythm headway are described by their mean headway. This procedure does not explicitly calculate transfer times but assumes that they depend on the headway. In other words, the co-ordination of the timetable is not considered. Usually, one assumes that the wait times at the boarding stop or at transfer stops is equal to half of the line's headway. Assignment based on lines guarantees acceptable assignment results only for urban areas with a dense network and short headways.
2. *Timetable-based* assignment considers the timetable of each transit line with its exact departure and arrival times [4]. Using a shortest path algorithm, it is possible to determine the "best" connection between two traffic zones for a particular departure time. For different times of departure, various "best" connections may be calculated differing in the used transit lines or transfer stops. Assignment based on timetable is the appropriate method when precise values for the transfer time and the service frequency are expected. This is especially important in rural areas or rail networks, where headways are long and the co-ordination of the timetable is important for the service quality.

A timetable-based search using a shortest path algorithm unfortunately has two weaknesses: it may require long computing time and it may not find all relevant connections. The example displayed in Figure 1 illustrates these weaknesses. The example network consists of a bus line and a train line. Passengers travelling from origin zone A to destination zone X may choose between direct connections provided by the bus and faster connections offered by the combination of bus and train:

- *Computation time*: To determine all connections with a shortest path algorithm, it is necessary to perform a search for each possible departure time at the origin stop within the examined time interval. For the example network, this requires three search steps starting at 6:10, 6:55 and 7:25. In a real life network, more than 100 different departure times at a stop are common (e.g. 10 min headway, two directions, 18 hours operating time $\rightarrow 2 \cdot 6 \cdot 18 = 216$ departures) resulting in long computation times as every departure time requires to build one shortest path tree. That this type of search routine may not be very efficient is illustrated in Figure 2, where 12 different departure times per hour at the origin stop O result in only four connections between stop O and stop D. Obviously, not every route search yields

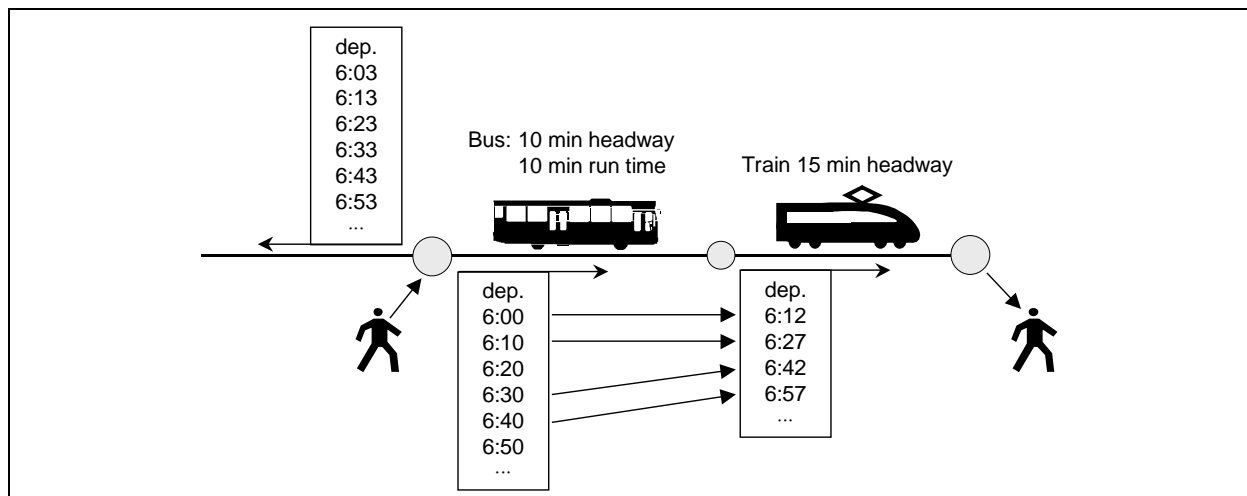
a new connection. Considering this phenomenon, it is possible to speed up the search process by limiting the number of departure times. This can be achieved either by randomly selecting a certain percentage of departure times or by searching only at predefined times, e.g. every 10 minutes. As an acceptable computing time may only be achieved through a significant reduction of departure times, this approach will usually fail to find all connections.

Figure 1: Example network



- Quality of results: Since the transit supply in some cases offers more than one connection for a given departure time (see example in Figure 1), a shortest path algorithm requires a definition of the "best" connection. For this purpose, it is common to apply an impedance function which increases the impedance of a connection for each transfer by means of a transfer penalty. A low penalty prefers time-minimal connections, while a high penalty gives priority to connections with lower transfer frequencies. Varying the transfer penalty, however, increases the computation time as each transfer penalty requires an individual search step. Identifying all five connections in the example network of Figure 1 requires six search steps: three search steps with a penalty of 0 minutes (→ connections 2, 3, 5) and three search steps with a penalty of at least 15 minutes per transfer (→ connections 1, 3, 4). The problem of defining an impedance function for the best connection increases with the introduction of fares. In a network with cheap low-speed trains and expensive high-speed trains, it is necessary to vary not only the transfer penalty but also the value of time. In such a network, even a connection which departs earlier and arrives later than an alternative connection may be attractive for some passengers, if it is cheaper or requires fewer transfers.

Figure 2: Departure times at an origin stop: 12 departure times per hour at stop O result in only four connections between O and D.



Extensive analysis of the shortcomings of a timetable-based search procedure using a shortest path algorithm and requirements of European railway companies (German Railways DB AG & DB Regio, Danish Railways DSB) to model railway networks on a national and European level, initiated the development of a new search routine for transit assignment, i.e. for a many-to-many search problem. After analysing several possible methods, an approach was chosen which is based on works by FRIEDRICH [5]. As his branch & bound approach, developed in 1994, uses a preprocessing step which requires more computer memory than conventional shortest path algorithms, the method could only now be adapted for big networks to work on standard PCs.

3 TRANSIT ASSIGNMENT WITH BRANCH & BOUND

3.1 Preprocessing

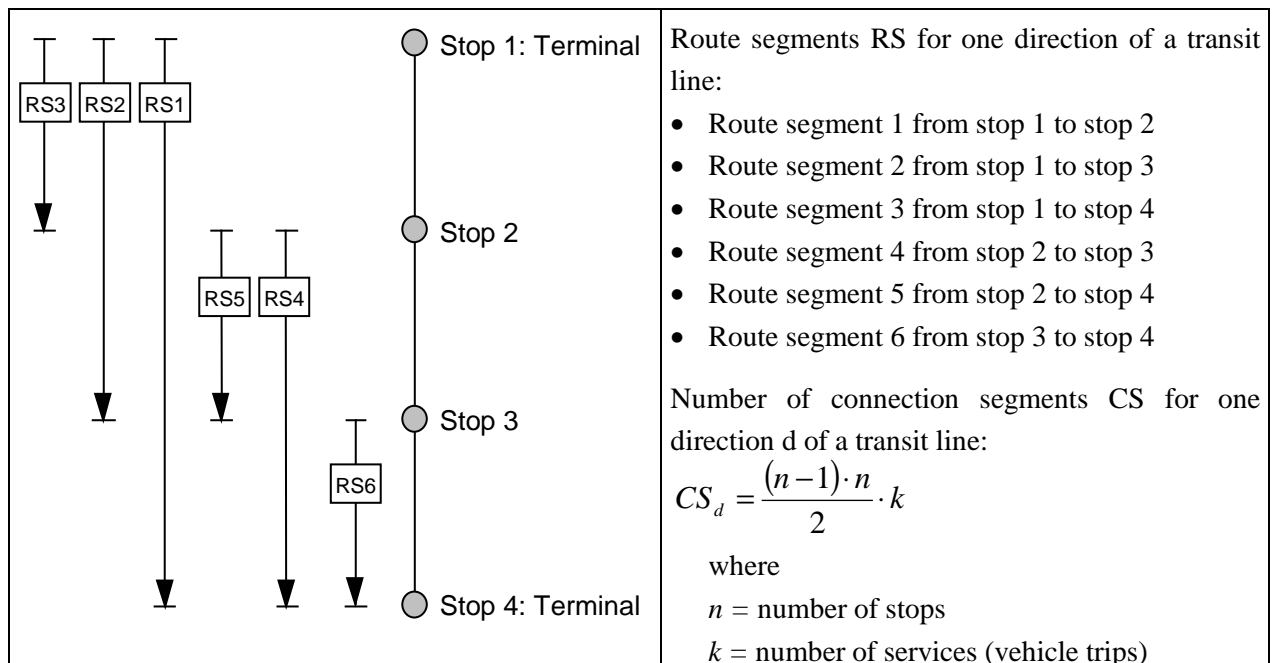
The objective of the preprocessing step is the generation of *connection segments*, which describe a part of a journey. These connection segments are concatenated to connections later in the connection search process. A connection segment represents either a walk or a transfer-free ride on one transit line. A connection made up of an access walk, a bus ride and an egress walk, for example, consists of three connection segments.

From the line route data, an initial step generates an array in which the network's *route segments* are stored (see Figure 3). A single route segment describes either

- a segment of a transit line between a boarding stop and an alighting stop or
- a single walk link or a sequence of walk links for access, egress or transfer walks.

Walk link sequences are calculated by applying a shortest path algorithm to the links of the network which are permitted for “transit walks”. A route segment is defined by the attributes initial stop node, terminal stop node, length and running time. It also points to the used transit line or the sequence of walking links respectively. The route segment array is sorted in an appropriate way to create connection segments in the next step.

Figure 3: Route segments and connection segments



Subsequently, *connection segments* (or connection legs) are calculated from the sorted list of route segments. A connection segment using a transit line is endowed with a departure and an arrival time, whereas connection segments using walk links do not possess specific times, since they are available to the traveller at any time.

The connection segments are again stored in a sorted array. Special data structures provide tools such as finding the next connection segment for a given node and point of time or traversing all connections originating from a fixed node. Figure 4 shows the route and connection segments for the example network of Figure 1, not considering the access and egress walk links, which are assumed to be 0 minutes.

Figure 4: Route segments and connection segments for the example network

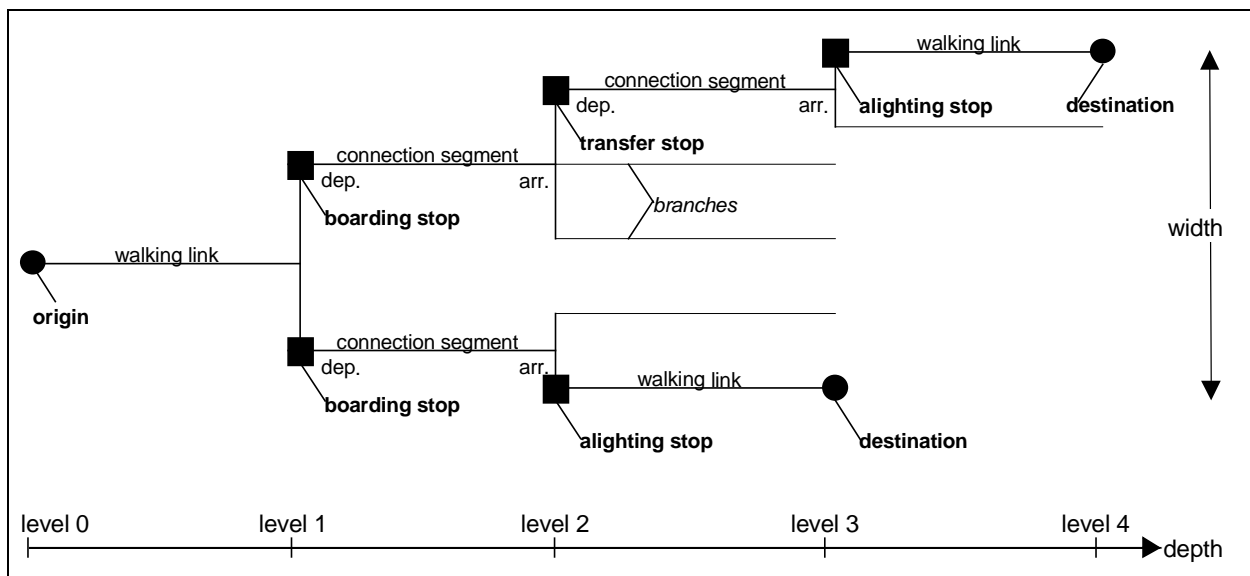
Route Segments				Connection Segments		
Route Segment No	From Node	To Node	Transit Line	Connection Segment No	Departure	Arrival
1	A-Village	Station	Bus 1	1	6:10	6:22
				2	6:55	7:07
				3	7:25	7:37
2	A-Village	B-Village	Bus 1	4	6:10	6:42
				5	6:55	7:27
				6	7:25	7:57
3	A-Village	X-City	Bus 1	7	6:10	6:55
				8	6:55	7:40
				9	7:25	8:10
4	Station	B-Village	Bus 1	10	6:22	6:42
				11	7:07	7:27
				12	7:37	7:57
5	Station	X-City	Bus 1	13	6:22	6:55
				14	7:07	7:40
				15	7:37	8:10
6	B-Village	X-City	Bus 1	16	6:42	6:55
				17	7:27	7:40
				18	7:57	8:10
7	Station	X-City	Train	19	6:25	6:41
				20	7:05	7:21
				21	7:45	8:01

3.2 Connection Search

A dynamic, i.e. time-dependent, multi-path algorithm is applied to determine all potential connections. This algorithm builds a connection tree (see Figure 5) which may contain several paths (connections) from the origin to a destination node. Since the tree's width largely depends on the service frequency of the transit lines, it may be much wider than a usual shortest-path tree. On the other hand, the use of entire connection legs as tree edges simplifies the tree's structure to a great extent and limits its depth by the maximum number of transfers. A typical value of this user-defined constant is 4 or 5. Since the algorithm's speed depends linearly on the tree's width but exponentially on its depth, the procedure behaves nicely in terms of computation time.

Figure 5 outlines the structure of the connection tree. The root of the tree is the centroid node of the origin zone and the only outgoing branch represents the access from the origin to the first transit stop.

Figure 5: Structure of the connection tree



The following parameters are used to describe the characteristics of a connection c :

- journey time $JT(c)$, departure time $DEP(c)$ and arrival time $ARR(c)$
- number of transfers $NT(c)$ and transfer time $TT(c)$
- fare $FARE(c) = \sum_{i \in I(c)} (SF_i + L_i(c) \cdot SD_i)$, where $I(c)$ is the set of all transit lines used by c . In this way, the fare of transit line i may include a fixed supplement SF_i and a distance-based fare $L_i(c) \cdot SD_i$, where $L_i(c)$ is the distance covered with transit line i .

Then, the total (search) impedance $IMP(c)$ of a connection is defined as

$$IMP(c) = a_1 \cdot JT(c) + a_2 \cdot NT(c) + a_3 \cdot FARE(c),$$

where a_1 , a_2 and a_3 are global user-defined constants.

Starting from the origin node, a branch & bound strategy is employed. Given a connection segment from the current tree level, all possible successors are considered. Thanks to the data structures established in earlier steps, these succeeding segments can be traversed without additional data access time.

Now, let $s_{x,y}^*$ be the currently processed connection segment starting at network node x and terminating at network node y . Let c_y^* be the new connection between the origin node and y formed by adding $s_{x,y}^*$ to connection c_x^* arriving at x . Finally, let C_y be the set of all known connections to y .

$s_{x,y}^*$ is inserted into the tree as an appendix of connection c_x^* , if and only if the following conditions hold:

- *Temporal suitability*: The connection segment $s_{x,y}^*$ departs from node x only after the arrival of connection c_x^* plus a minimum transfer wait time $MINTWT$, and before a user-defined maximum transfer wait time $MAXTWT$ has elapsed:

$$DEP(s_{x,y}^*) - ARR(c_x^*) \in [MINTWT; MAXTWT]$$

- *Relevance*: There is no known connection $c \in C_y$ such that

$$DEP(c) \geq DEP(c_y^*) \text{ and } ARR(c) \leq ARR(c_y^*) \text{ and } IMP(c) \leq IMP(c_y^*) \text{ and } NT(c) \leq NT(c_y^*).$$

In other words, if c_y^* has the property that for all known connections c to y , c_y^* departs later or arrives earlier or has a lower impedance or a lower number of transfers than c , c_y^* will be inserted. To facilitate comparisons between elements of C_y , these sets are stored in lists which are sorted by arrival time.

- *Tolerance constraints*: None of the following rules is violated:

$$IMP(c_y^*) \leq b_1 \cdot \min_{c \in C_y} IMP(c) + b_2$$

$$JT(c_y^*) \leq d_1 \cdot \min_{c \in C_y} JT(c) + d_2$$

$$NT(c_y^*) \leq e_1 \cdot \min_{c \in C_y} NT(c) + e_2$$

$$NT(c_y^*) \leq MAXNT,$$

where all b_i , d_i and e_i are user-specified global tolerance parameters and $MAXNT$ is the also user-defined bound for the number of transfers within a connection.

- *Loops*: Transfers *within* a transit line are only allowed for lines containing a loop, provided that the passenger can save time by boarding an earlier service trip at the intersection node of the route loop.

A tree level is traversed completely before connection segments of the next level are considered. For this purpose, we hold two queues storing unprocessed segments of the current and the next level respectively. The procedure terminates when the queues are empty or the maximum number of transfers is reached.

With regard to the accessibility of the algorithm's outcome, some search information is stored for each of the network nodes. It comprises the shortest journey time, the shortest walking time, the minimum number of transfers and the minimum fare taken over all connections to that node established up to the present. Other values describing the overall characteristics of a connection are passed through all corresponding branches.

3.3 Connection Choice

The set of connections stored in the connection tree needs to be re-evaluated in order to identify and delete illogical connections (e.g. equal departure time with later arrival time and more transfers), which have not been rejected by the search algorithm.

In the search algorithm, tolerance constraints like $JT(c_y^*) \leq d_1 \cdot \min_{c \in C_y} JT(c) + d_2$ (see above) are applied to a newly-found connection c_y^* . In this way, those conditions are checked *at every intermediate node* y of a connection. Therefore, the tolerance parameters b_i , d_i and e_i should not generate too strict constraints, since too many connections would be discarded as a result.

The connection choice step enables the user to specify stricter conditions for the *entire* connection. This is realised by the same set of tolerance constraints, where this time the constants b_i , d_i and e_i can be chosen smaller. Typical values of d_1 , for instance, may range between 1.2 and 1.8 to reflect the fact that a traveller is not willing to use connections significantly slower than the fastest available.

3.4 Connection Split

The final assignment step simulates the passengers' choice among the remaining set of connections. It is assumed that their decisions are based on the perceived journey time, the difference between the desired and the actual departure time and the fare.

The perceived journey time is a user-defined function of the actual journey time, the transfer time and related quantities. A typical definition is

$$PJT(c) := JT(c) + 2 \cdot TT(c) + 2 \cdot NT(c).$$

A temporal distribution of the travel demand serves as the input for the connection split. For practical purposes, this distribution is discretised in equidistant time intervals (e.g. hours) with constant travel demand. For such a time interval a , let $C(a)$ be the set of the connections departing in this period. The formula for the impedance of a connection c in time interval a is

$$IMP_a(c) := q_1 \cdot PJT(c) + q_2 \cdot U_a(c) + q_3 \cdot FARE(c),$$

where $U_a(c)$ is the temporal utility for passengers departing in time interval a and q_1, q_2, q_3 are user-set global constants. Note that the concept of impedance introduced for the search is re-adjusted to the requirements of the connection split. The temporal utility function $U_a(c)$ models the difference between the riders' desired and the actual departure time of c . Usually, $U_a(c) = 0$ for all $c \in C(a)$ and $U_a(c) > 0$ for all $c \notin C(a)$, where the graph of $U_a(c)$ increases monotonously outside a .

Denote by C the set of all connections. The part of the travel demand of time interval a assigned to connection $c \in C$ will be calculated by means of a utility function which also takes into consideration the "independence" of c . The independence of a connection $c \in C$ is defined as

$$IND(c) := \frac{1}{\sum_{c' \in C} f_c(c')} = \frac{1}{1 + \sum_{c' \in C, c' \neq c} f_c(c')},$$

where f_c is an appropriate non-negative evaluation function, for which $f_c(c) = 1$ and $f_c(c') \leq 1 \forall c, c' \in C$. The purpose of f_c is to model the impact of other connections on c . The general rule is: the greater the similarity of two connections, the higher the impact on one another. If there are differences in terms of perceived journey time and fare, the superior connection will exert a stronger influence on the inferior one than vice versa. This concept of asymmetry is indispensable for $IND(c)$ to perform properly. f_c might be defined as follows:

$$f_c(c') := \left(1 - \frac{x_c(c')}{s_x}\right)^+ \cdot \left(1 - \gamma \cdot \min\left\{1, \frac{s_z \cdot |y_c(c')| + s_y \cdot |z_c(c')|}{s_y \cdot s_z}\right\}\right),$$

where $x_c(c') := (|DEP(c) - DEP(c')| + |ARR(c) - ARR(c')|)/2$ is the temporal similarity of c and c' , $y_c(c') := PJT(c') - PJT(c)$ the advantage of c with respect to the perceived journey time and $z_c(c') := FARE(c') - FARE(c)$ the corresponding value with respect to the fare. s_x , s_y and s_z set the range of influence of the three variables and γ is a global parameter. s_y and s_z depend on the signs of $y_c(c')$ and $z_c(c')$ in order to model the asymmetry mentioned above. Among other aspects, such an evaluation function penalizes short lags between departure times meaning that connections with identical or similar departure times are assigned a lesser independence.

For the final assignment step, define the Box-Cox transformation as the following parameterized function of $IMP_a(c)$:

$$b^{(t)}(IMP_a(c)) := \begin{cases} \frac{IMP_a(c)^t - 1}{t} & \text{if } t \neq 0 \\ \log(IMP_a(c)) & \text{if } t = 0. \end{cases}$$

Denote by $P_a(c)$ the portion of the demand $DEM(a)$ of time interval a which is assigned to connection c . Using the MNL model enhanced by the new attribute of independence, $P_a(c)$ is defined as

$$P_a(c) := \frac{e^{-\beta \cdot b^{(t)}(IMP_a(c))} \cdot IND(c)}{\sum_{c \in C} \left(e^{-\beta \cdot b^{(t)}(IMP_a(c))} \cdot IND(c)\right)} \cdot DEM(a),$$

where β is a positive parameter. Now, the effect of the concept of independence can be made clear quite easily. Consider the following simple example where $DEM(a) = 120$ passengers are faced with several connections in time interval $a = [8:00, 9:00]$. Since all connections depart within a , it is assumed that $U_a(c) = 0$ for all $c \in C(a)$, meaning that the travellers are indifferent as to the precise departure time within a . For simplicity, let $q_1 = 1$ and $q_2 = q_3 = 0$. The evaluation functions used are the f_c defined above. Letting $\beta = 2$ and $t = 0$ gives the standard Kirchhoff distribution.

The first example (Figure 6) shows three identical connections with a fixed headway. All connections are independent, so that $IND(c)$ is always equal to 1.0 and each connection is assigned the same amount of passengers.

Figure 6: Connection split example 1 – three connections with a fixed headway

c	$DEP(c)$	$PJT(c)$	$IND(c)$	$P_a(c)$ not using $IND(c)$	$P_a(c)$ using $IND(c)$
1	8:15	20 min	1.00	40	40
2	8:30	20 min	1.00	40	40
4	8:45	20 min	1.00	40	40

In the second example (Figure 7) a further connection is added which departs at the same time as connection 2. Here the use of $IND(c)$ causes the split to assign just as many passengers to connections 2 and 3 as to each one of the connections 1 and 4. In this way, it is taken into account that 2 and 3 are in fact only *one* connection.

Figure 7: Connection split example 2 - insertion of a connection with simultaneous departure time

c	$DEP(c)$	$PJT(c)$	$IND(c)$	$P_a(c)$ not using $IND(c)$	$P_a(c)$ using $IND(c)$
1	8:15	20 min	1.00	30	40
2	8:30	20 min	0.50	30	20
3	8:30	20 min	0.50	30	20
4	8:45	20 min	1.00	30	40

The third example (Figure 8) shows the impact of an express connection on the distribution. Without $IND(c)$, the express connection 3 detracts equally many passengers from all other connections, although its impact on connection 1 may be assumed to be weaker than on the neighboring connections 2 and 4. The figure shows that this aspect *is* considered when $IND(c)$ is used.

Figure 8: Connection split example 3 - insertion of an express connection

c	$DEP(c)$	$PJT(c)$	$IND(c)$	$P_a(c)$ not using $IND(c)$	$P_a(c)$ using $IND(c)$
1	8:15	20 min	1.00	25	27
2	8:30	20 min	0.86	25	23
3	8:40	15 min	0.94	45	46
4	8:45	20 min	0.86	25	23

4 APPLICATION AND CONCLUSION

The main motivation for the development of the presented transit assignment method were complaints about computation time. For example, transportation planners of German Railways using the timetable-based transit assignment implemented in the transportation model VISUM [6] came up with computation times of more than 24 hours for the German train network. Reasonably, they asked for a transit assignment which would run overnight, ideally not just for the German network but also for a European network.

Figure 9 shows the main parameters describing the German train network model. For this network, a search procedure using a shortest path algorithm builds a shortest path tree for one departure time at an origin node in about one second on a 500 MHz Pentium 3 computer. This is a good response time for an interactive connection search. Applying this search procedure to an assignment problem of the given size, however, results in a total computation time of 32.5 hours, if every departure time is considered. Even a reduced search with only 4 randomly selected departure times per hour and 18 hours operating time ($= 4 \cdot 18 = 72$ departures per zone) would require approximately 10 hours of computing time. In contrast, the branch & bound search, which is now also implemented in VISUM, completes the assignment about 20 times faster in only 1.6 hours. Instead of computing an average of 220 shortest path trees for every zone, the branch & bound search builds just one connection tree per zone. This approach of constructing only one connection tree from preprocessed connection segments obviously speeds up the search procedure. At the same time, it expects more computer memory for storing the route and connection segments. The German train network with 2.7 million connection segments requires an extra of about 50 MByte RAM. This, however, does not create a problem for current standard PCs with 256 or 512 MByte of RAM.

Beside the significant gains in computing time, the branch & bound search increases the quality of results as it tends to find more connections. This applies especially for cases, where two connections have the same departure times or where passengers can choose between cheaper low-speed trains and more expensive high-speed trains. Here, a shortest path search always has to cope with the problem that it will identify either the fastest or the cheapest or the minimum transfer connection. The branch & bound search, on the other hand, behaves more robustly in such cases. The introduction of flexible tolerance parameters in combination with the multi-path connection tree permits to host connections with varying indicators for time, cost and number of transfers.

Including private transport links in the preprocessing step allows a straightforward extension of the branch & bound search to park & ride traffic. For this purpose, the preprocessing step would also have to determine private transport route segments between traffic zones and park & ride transit stops as a sequence of road links. Private transport route segments could then be included in the search procedure just like walking links, but with the additional constraint that they only occur as the first or the last segment. Considering that returning travellers need to collect their private transport vehicle at the same stop where they parked it, an intermodal park & ride assignment additionally would need to store the most likely park & ride stop for every o-d pair.

Figure 9: Performance of timetable-based assignment

German train network model	
no. of traffic zones	531
no. of transit stops	6,300
no. of lines with identical line route and running time	3,040
no. of vehicle trips (services) per day	31,600
mean no. of departure times per traffic zone	220
no. of route segments	273,000
no. of connection segments	2,737,000
Computing time* with shortest path search	
• mean computing time for one departure time (= one shortest path tree)	1 sec
• mean computing time for one zone	220 sec
• total computing time	32.5 h
Computing time* with branch & bound search	
• computing time for preprocessing	20 sec
• mean computing time for one zone	11 sec
• total computing time	1.6 h
* Computing time using a 500 MHz Pentium 3 Computer	

5 REFERENCES

1. Ortúzar, J. D., Willumsen, L. G.: *Modelling Transport*, Wiley, Chichester, 1990.
2. Schnabel, W., Lohse, D.: *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung*, Volume 2, Verlag für Bauwesen, Berlin, 1997.
3. Spiess, H., Florian, M.: Optimal Strategies: A New Assignment Model For Transit Networks, in *Transportation Research*, 23B(2), 82-102, 1989
4. Friedrich, M.: Modelling Public Transport - A European Approach. *Preprint CD-ROM of 78th Annual Meeting*, Transport Research Board, Washington, 1999.
5. Friedrich, M.: Rechnergestütztes Entwurfsverfahren für den ÖPNV im ländlichen Raum ("Computer assisted design of public transport systems in rural areas"), in *Schriftenreihe des Lehrstuhls für Verkehrs- und Stadtplanung*, Volume 5, Technical University of Munich, 1994.
6. PTV AG: VISUM, <http://www.english.ptv.de/produkte.htm>, July 2000