An A* label-setting algorithm for multimodal resource constrained shortest path problem

Tai-Yu Ma

© 2013 The Authors. Published by Elsevier Ltd.
Selection and/or peer-review under responsibility of Scientific Committee.

Keywords: shortest path; resource constraint; multimodal; label setting.

1. Introduction

The multimodal resource constrained shortest path problem (MCSP) consists of finding a least travel time path with single/multiple resource(s) and mode chain constraints. The problem is an extension of constrained shortest path problem (CSP) in the context of multimodal network. The development of multimodal constrained shortest path algorithm is an important issue since it can be applied for multimodal route advisory system, activity-chain route planning in activity-based travel demand analysis. However, past studies focused either on general shortest path problems or road constrained shortest path problems. There still lack efficient and effective algorithms for solving the MCSP problem.

* Corresponding author.
E-mail address: tai-yu.ma@ceps.lu

© 2013 The Authors. Published by Elsevier Ltd.
Selection and/or peer-review under responsibility of Scientific Committee.
doi:10.1016/j.sbspro.2014.01.066
The CSP problem on unimodal network has been widely studied in the past. It is generally formulated as a 0-1 integer programming problem with edge-based decision variables. Handler and Zhang (1980) formulate the problem as a dual integer programming problem and propose a two-stage procedure for its resolution. The first step consists in finding the lower bound of the Lagrangian relaxation problem and, at the second step, closing the gap by utilizing k shortest paths to identify the first shortest path which satisfies its resource constraints. Beasley and Christofides (1989) propose an integer programming formulation and use Lagrangian relaxation technique to solve it. A subgradient method is proposed to find lower bound and then a tree search procedure is applied to find optimal solution. Different with integer programming method, Aneja et al. (1983) propose a two-stage label setting approach for multiple resource constraint problems. At the first stage, infeasible nodes and edges violating resource constraints are pruned. At the second stage, a set of labels for resource constraints are associated with nodes and a Dijkstra’s algorithm, starting from its source, progresses the search by checking non-dominated labels of outgoing nodes until the target node is reached. Ziegelmann (2001) proposes a path-based integer linear programming formulation of the CSP problem and solved its Lagrangian relaxation problem by a two-stage procedure. The author proposes an efficient hull approach to obtain lower bound and then, at the second stage, using different gap-closing methods to find optimal solutions. However, the methods are designed for unimodal CSP problems. To the best of our knowledge, the MCSP problem has not yet been studied in the literature.

For the multimodal shortest path problem without resource constraint, previous studies focused on computing the shortest path based on labeled-Dijkstra’s algorithm to satisfy mode sequence constraints. Each edge/node of network is associated with related mode label susceptible for verifying the mode chain constraint of path. The mode chain constraint is modeled by non-deterministic finite automata to progress the path searching under admissible state transition for next outgoing node (Mendelzo and Wood, (1989); Barrett et al. (2000)). Sherali et al. (2003) apply non-deterministic finite automata for solving time-dependent label constrained shortest path problem. The proposed algorithm is based on Dijkstra’s algorithm by checking the feasibility of mode-label constraints on its path searching process. However, it cannot be applied for solving multiple resource constrained problem. Different with the automata-based method, Lozano and Storchi (2001, 2002) propose a label correcting algorithm to find viable minimum cost paths in a multimodal network under constrained mode chain and number of transfers. The viability of mode sequence is verified by specifying state transitions for viable paths in path constructing process. The obtained viable paths contain Pareto-optimal paths that have minimum expected travel time and under modal transfers upper limit.

In this study, we present a label setting algorithm combined with A* algorithm acceleration technique for solving the MCSP problem (Hart et al., 1968; Ikeda et al., 1994). The algorithm utilizes a priority queue to propagate path search in checking a set of labels progressively generated in the search process. Each label is a vector to record consumed resources at current state. The labels on a node are ordered by a key evaluated by a goal-directed evaluation function. By eliminating the labels having been dominated, the label list to be considered can be kept small and hence reduce the computational time. For mode chain constraint, a non-deterministic finite automata is used to check the constraint with the cost of time complexity of O(1). The main contribution of this work is to propose a fast label-based algorithm, combing some state-of-the-art acceleration techniques to fast compute multiple resources constrained multimodal path on a multilevel graph.

The rest of the paper is organized as follows. In Section 2, we propose a multilevel directed graph for multimodal transportation network modeling. The basic definitions of labeled graph and non-deterministic finite automata are introduced for characterizing the multimodal network. Then we formulate the MCSP problem by 0-1 integer programming. In Section 3, we present the proposed label setting algorithm and accelerating technique based on the A* algorithm and Access-Node routing technique. Section 4 presents the computational study on static and dynamic realistic networks drawn from Lorraine network (France). Finally, the conclusion is drawn and future extensions are discussed.
2. Multilevel multimodal network and the constrained shortest problem

In this section, we first introduce the multilevel directed graph for modeling multimodal network routing problem. To take into account the modal constraints on the path finding process, the finite automata and the resulting product graph are incorporated. Then, we formulate the MCSP problem as a 0-1 integer programming problem on the product graph. A simple example is illustrated to explain the main idea of the proposed model for multimodal path modeling.

2.1. Multilevel multimodal network

Let $G(V, E, M)$ be a multi-level directed graph representing a multimodal network, where $V$ is the set of nodes, $E$ the set of links (edges) and $M$ the set of modes. We distinguish two subsets of modes: private modes (foot and car) and public modes (metro, tram, bus and train). Let a subgraph $G_m = (V_m, E_m)$ denote the unimodal network of mode $m$, where $V_m$ and $E_m$ is the set of nodes and the set of links of mode $m$, respectively. We denote a link $e_{ij}$ from node $i$ to node $j$. Each link is characterized with travel time and a vector of resource consumption. Note that the resource consumption can be monetary cost or mode transfer counting label etc. We denote the road network as reference network, where each node is called reference node characterized by its x-y coordinates. Given a reference node $n$, a set of access-nodes $V(n)$ is defined as the set of mode-specific nodes accessible for the reference node $n$ on a public network via transfer links. We call the corresponding access-nodes as station nodes, representing an access point to public transportation services.

Each subnetwork of mode $m$ is represented at one level and interconnected by a set of transfer links. The transfer links are a set of walking links connecting: 1) a reference node and its access-nodes, and 2) the access-nodes located at the same x-y coordinates. Let $E^T_m$ denote the set of transfer links connecting the subnetwork of mode $m$ with other subnetworks via reference nodes/access-nodes. The multimodal graph is then the union of all subnetworks and the set of transfer links $G(V, E, M) = \bigcup G_m(V_m, E_m) \cup E^T_m$.

Given available page limit, we give a brief description of road and public transportation network, and their connections as follows.

- Level 1 (road network): the road network can be used by both pedestrians and cars, or only by one mode of them. Based on its modal viability, related travel time function is associated on each link and evoked according its arrival mode state on the route searching process. Such implementation can reduce half of the size of the multimodal network.

- Level $k$ (unimodal public transportation network): each unimodal public transportation network of mode $k$ is represented by a directed subgraph $G_k$. Two sets of nodes are distinguished: the set of station nodes and that of line nodes. The travel time on a boarding/alighting link is the average transfer times between a station and the boarding point of vehicle. We assume operation services are based on fixed schedule timetables. The travel time on a line link is then calculated as waiting time (differences between arrival times at line node and departure times of next vehicle) plus in-vehicle travel times on the link based on a piecewise linear function over a finite set of discrete departure time of vehicle.

Let us give an example of a multilevel multimodal network as shown in Figure 1. The road network is connected with a number of public transportation subnetworks via related access-nodes (stations). The unimodal stations are interconnected via transfer links requiring nonzero transfer times between the subnetworks. Note that from node 13 to node 16 (Figure 1) on the train network, there is a direct connection without stops between them (express train). The transfer links inter-connect road and public transportation network via access-nodes (station
nodes. For imposing mode sequence constraints on the multimodal path searching process, we introduce in the following the concept of the finite automata and product graph for this issue.

Fig. 1. A multilevel multimodal network example

2.2. Finite automata and product network

We recall here basic concepts of non-deterministic finite automata and product graph for the multimodal network routing problem. The reader is referred to Barrett et al. (2000) and Pajor (2009) for more detailed description.

Definition 1. Language and label

Let \( G = (V, E) \) be a weighted direct graph where \( V \) is the set of nodes and \( E \) the set of arcs. Let \( \Sigma \) denote a finite set of symbols (mode-related labels) called alphabet. A sequence of symbols is called a word. We define a language \( \mathcal{L} \) as a finite set of admissible words based on \( \Sigma \). For each \( e \in E \), a label \( \sigma \in \Sigma \) is associated with it. We call the labeled graph as \( \xi \)-labeled graph \( G_\xi \). The sequence of labels on a path constitutes a word \( w \).

Definition 2. Finite automata

A non-deterministic finite automata (called automata hereafter) is defined as \( A = (Q, \Sigma, \delta, s_0, F) \) where \( Q \) denotes a finite set of states, \( \Sigma \) a alphabet, \( \delta \) a transition function \( \delta : Q \times \Sigma \to Q \), \( s_0 \in S \) an initial state with \( S \subseteq Q \). \( F \subseteq Q \) is a set of final states. Given a node \( i \) with current state \( q_i \), an admissible state is drawn from the transition function \( \delta(q_i, \sigma), \forall \sigma \in \Sigma \). Note that \( q_i \) is the mode state at node \( i \) and \( \sigma \) is mode label associated for an edge. The transition function permits the mode state change from one state to another via its used edge. A language \( \mathcal{L} \) is accepted by the automata \( A \) if and only if the words in \( \mathcal{L} \) are accepted by the automata \( A \).

Definition 3. Product graph

Given a \( \xi \)-labeled graph \( G_\xi \) and a finite automata \( A = (Q, \Sigma, \delta, s_0, F) \), a product graph \( \tilde{G} = (\tilde{V}, \tilde{E}) \) is composed of a set of product-node \( \tilde{V} \) with its component \( \tilde{v} = (v, q), v \in V, q \in Q \), and a set of product-edge \( \tilde{E} \) with its
element $\tilde{e} = (\tilde{v}, \tilde{v}_j)$, $i \neq j$, $\forall \tilde{v} \in \tilde{V}$, where there exists an admissible transition in the automata such that $q_j \in \delta(q_j, \sigma)$. An example is shown in Figure 2.

The label constrained shortest path problem consists of finding shortest paths $p = (\tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_n)$ from its source to its destination such that the mode sequence on the path satisfies the associated finite automata $A$ of the graph. The travel time of path is the sum of travel times on the edges of $p$. As shown in Barrett et al. (2000), finding a shortest path in $\tilde{G}$ can be solved by applying Dijkstra’s algorithm on the product graph. As explicitly implementing the product graph is space and time expensive, previous study (Pajor, 2009) proposed an efficient technique based on Dijkstra’s algorithm which incrementally checks the next outgoing edges complying with the automata. Note that this checking is very efficient which needs only match the state transition table with its current state and the querying link label. The time complexity is $O(1)$. 

![Diagram of label-transition graph $G_k$ and its transition table](image)

**Fig. 2.** Label-transition graph $G_k$ (left) and its transition table (right)

### 2.3. Integer programming formulation of the MCSP problem

Consider a MCSP problem on a product graph $\tilde{G} = (\tilde{V}, \tilde{E})$ with the language $\mathcal{L}$ and the automata $A = (Q, \Sigma, \delta, S, F)$. Let $c_{ij}$ be the travel time of product-link $(i, j) \in \tilde{E}$, $r^k_{ij}$ is the $k$-th resource consumption over link $(i, j)$, $k=1,2,\ldots,K$. $\lambda^k$ is the $k$-th resource limit. Note that we assume all $c_{ij}$, $r^k_{ij}$ and $\lambda^k$ are non-negative. The 0-1 integer programming formulation of the MCSP problem can be defined as

$$\text{Min} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

s.t. $\sum_{j \in V} r^k_{ij} x_{ij} \leq \lambda^k$, $\forall k = 1,2,\ldots,K$  

$$\sum_{j \in V} x_{ij} = \sum_{j \in V} x_{ji}, \forall i \in \tilde{V} \setminus \{s, u\}$$

where $\tilde{V} = \{s, u, v_1, v_2, \ldots, v_n\}$.
\[ \sum_{j \in V} x_{ij} = 1 \quad \text{(4)} \]
\[ \sum_{j \in \tilde{V}} x_{ju} = 1 \quad \text{(5)} \]
\[ x_{ij} \in \{0,1\}, \forall i, j \in \tilde{V} \quad \text{(6)} \]

The objective function (1) states the path travel time to be minimized. Equations (2) state the path resource consumption is constrained by its resource limits. Equations (3)-(6) ensure the acyclic path connecting its source \( s \) and its target \( u \) on the product graph \( G \) satisfying the user-defined automata \( A \).

The MCSP problem can be solved by classic 0-1 integer programming methods with the price of constructing corresponding product graph explicitly. However, in real application, the memory space needed to store the graph would be explosive with \(|G| \times |A|\) with \( G \) the directed graph and user-defined finite automata (Pajor, 2009). Hence we proposed a label setting algorithm based on implicitly checking the admissibility of transition state in the automata for outgoing edges. To accelerate the query speed, an accelerating technique based on A* algorithm and Access Node routing is incorporated in the solution algorithm. The solution algorithm is composed of two steps: a pre-processing step and the algorithm based on Dijkstra’s label setting algorithm. The pre-processing step first generates a reduced graph by eliminating unnecessary nodes and arcs. The obtained reduced graph is then used for computing the landmark distance table with respect to travel time in the objective function.

3. A* label setting algorithm

The proposed algorithm operates on a \( \Sigma \)-labeled product graph \( \tilde{G}(\tilde{V}, \tilde{E}) \) for the language \( \mathcal{L} \) and the finite automata \( A := (Q, \Sigma, \delta, s_0, F) \). The automata describe the language \( \mathcal{L} \) by associating nodes and links of the graph with mode-related labels. The central idea of the proposed algorithm is based on a resource label setting and pruning process in order to find resource constrained shortest paths. The resource label stores the distance (travel time) and consumed resources from the source node to its target node. In the path searching process, the resource labels are progressively created and associated on its reached node in an ascending order according to a key. By eliminating the dominated-labels (described later) stocked on the same node, inefficient path can be pruned. Each non-dominated label represents an efficient path from a source node to currently reached node. The path can be reconstructed by back-tracking the sequence of predecessors of labels to the source of path. Note that the label setting algorithm is known as Dijkstra’s algorithm for non-resource constrained shortest path problem. However, for our resource constrained shortest path problem, one has to examine all potential labels of reached nodes. As the number of labels to be examined is extremely large, the problem is well-known as a NP-hard problem (Garey and Johnson, 1979). The proposed label setting algorithm is then based on an effective dominance checking process to reduce its computational time.

As mentioned above, it would be inefficient to directly search the shortest path by the label setting algorithm when applying on realistic network. For this issue, we apply the graph reduction technique for fast resource feasibility check (Aneja et al., 1983). The proposed algorithm is then composed of two steps. First a reduced graph \( \tilde{G}(\tilde{V}', \tilde{E}') \) is obtained by pruning infeasible nodes and links which violate at least one of resource constraints when passing through these nodes/links on the minimum resource path. The second step is based on the A*-based label setting algorithm to efficiently explore adjacent nodes according to a key evaluating approximately the distance (travel time) to the target.

In the path searching process, to efficiently select next adjacent nodes to explore, we utilize two techniques to reach a fast pruning of inefficient labels. First, a priority queue of labels is maintained according to a key of approximate distance (travel time) to the target node. Second, to prune inefficient labels, we associate with the
nodes with a duplicate of non-dominated labels in an increasing order of keys by pruning dominated labels. Compared with the native approach which explores next adjacent nodes according to the distance from the source, the A*-based acceleration technique can significantly reduce the number of explored nodes by investigating first the paths with smaller estimated travel time. As our algorithm operates on a directed graph with non-negative time-dependent travel time function, the bi-directional search acceleration technique is not convenient for our problem. This is because arrival time at target node cannot not be estimated in advance, resulting in the inverse shortest path computation difficult (Pajor, 2009).

In the following, we give the definitions of terms used for the proposed algorithm and then present the A*-label setting algorithm.

**Definition 4**: A label \( l_i \) associated at node \( i \) is a pair \((C_i, R_i)\) with a key \( C_i \) (e.g. travel time) and a vector of consumed resources \( R_i = (R_{i1}, R_{i2}, \ldots, R_{iK}) \) from a source node \( s \) to node \( i \). Given two labels at node \( i \), \( l_1 = (C_1, R_1) \) and \( l_2 = (C_2, R_2) \), we say \( l_1 \) dominates \( l_2 \) if \( C_1 \leq C_2 \) and \( R_1 \leq R_2 \), i.e., \( R_{i1} \leq R_{i2} \), \( \forall k \in \{1,2,\ldots,K\} \), and \( R_{i1}, R_{i2} \) are not all equal for all \( k \). Let \( L_i \) be the set of ordered non-dominated labels associated at node \( i \) by increasing order of keys. We denote \( k \)th label in \( L_i \) as \( l_{i,k} \).

Note that each label is associated (stocked) with a node and store the information concerning its predecessor, travel time and consumed resources from the source of path to the reached node.

**Definition 5**: Let \( \mathbf{V}^r \subset \mathbf{V} \) be a set of road nodes and station nodes of public transportation subnetworks, called query nodes. Let \( f(i, j) \) and \( g(i, j) \), \( k=\{1,2,\ldots,K\} \) be the minimum travel times and minimum consumed resource \( k \) between node pair \((i, j)\), respectively. We denote a vector \( g(i, j) \) with its element \( g_k(i, j) \), \( \forall k\in\{1,2,\ldots,K\} \) and a comparative operator \( \succ \) for comparing two vectors of same dimension. We say \( A \succ B \) is true if and only if there exists at least one element in \( A \) such that \( a_i > b_i \) and for all others elements \( a_j \geq b_j \).

The steps of the A*-label setting algorithm for the MCSP problem is illustrated below. Our implementation of priority queue is based on the binary heap which can get an approximate complexity of \( O(m + n \log n) \). The reader is referred to Festa (2006) for other implementation techniques.

### Table 1. A* label setting algorithm

| **Input and pre-processing:** |
|---|
| 1. Input a source \( s \), a target \( u \), a departure time instant \( t_0 \), an initial state \( s_0 \), a language \( \mathcal{L} \) and an automata \( A := (Q, \Sigma, \delta, s_0, F) \). The resulting labeled product graph is \( \tilde{G}(\mathbf{V}, \tilde{E}) \). |
| 2. For all pairs of query nodes (Definition 5), create the minimum travel time and resource query tables with respect to each type of resources. Compute the minimum travel time query table for all public transportation nodes to all station nodes. |
| **Output:** A multimodal shortest path between \((s, u)\) with departure time \( t_0 \) satisfying the automata \( A \). |

| **Step 1** Apply the graph reduction procedure (Aneja et al., 1983) and obtain the reduced graph \( \tilde{G}'(\mathbf{V}', \tilde{E}') \), go to Step 2. |
| **Step 2** A*-label setting algorithm |
3: Set $A=\{s\}, D=\widetilde{V} \setminus \{s\}$, set $l_{s1} = (0,0)$, a priority queue of labels $Q = \emptyset$
4: $Q \leftarrow l_{s1}; L_s \leftarrow l_{s1}$
5: while $Q \neq \emptyset$ do {
6: Pop up the label with smallest key in $Q$. Let the pop-up label be $l_{ih} = (C_{ih}, R_{ih})$
7: where its associated node is $i$; If $i = u$, Stop; Otherwise do {
8: for all outgoing links $e_{ij}$ do {
9: if $e_{ij} \in E_i$ and states $q_i \in \delta(q_i, \sigma_{ij})$ do {
10: if not $(R_{ih} + r_{ij} + g(j,u) \geq \lambda)$ {
11: create a new label $l_{js} = (C_{js}, R_{js})$ with 
12: $C_{js} = C_{ih} + c_0 + f(j,t)$, $R_{js} = R_{ih} + r_{ij}$
13: pred($l_{js}$) = $l_{ih}$;
14: dominance check procedure ($l_{js}, L_j$)
15: }}}
16: Dominance check procedure ($l_{js}, L_j$)
17: if $L_j = \emptyset$ then $L_j \leftarrow l_{js}$; $Q \leftarrow l_{js}$
18: else {
19: for $k=1$ to $|L_j|$ do {
20: if $(C_{js} \leq C_{jk})$ do {
21: if $l_{js}$ is dominated by $l_{jk-1}$ then delete $l_{js}$
22: else do {
23: Inset $l_{js}$ at the position before $l_{jk}$; $Q \leftarrow l_{js}$
24: for $m = k$ to $|L_j|$ do {
25: if $R_{js} \leq R_{jm}$
26: delete label $l_{jm}$
27: else break;
28: }}}}  
29: } 
30: } 
31: } 
32: } 
33: For the pre-processing step, the state transition table of the automata needs to be defined to specify admissible mode state changes in the path construction process. Each link is associated with related mode label and static or dynamic link travel time functions. For all OD pairs, the lower bound is computed with respect to travel time and each resource. In the graph reduction step, we check first whether the minimum resource path is feasible. If not, there is no feasible path satisfying its resource constraints. Otherwise, check whether the minimum consumed resource by passing node $i$ is feasible. If it is not feasible, delete the node and its incoming and outgoing links.
For the label setting algorithm, initially a new label \( l_0 = (0,0) \) is inserted in the priority queue \( Q \) and associated to the source \( s \). Each label is characterized by a key which contains two terms: travel times from the source to its reached node \( i \) and travel time estimation from \( i \) to the target \( u \). We use the x-y coordinate of road node to estimate the lower bound. As for the station node in public transportation network, the computation of the lower bound is calculated by two steps. First, an efficient k-d-tree search algorithm is applied to get \( k \)-nearest stations (Access Node) to the target. Then the approximated travel time from a public transportation node to the target is the minimum of the shortest travel time via these \( k \)-nearest stations to the target.

4. Computational study

In this section, we present the computational results of the A*-label setting algorithm on 4 instances drawn from Lorraine network (France). The algorithm is implemented on C++ on a Dell Latitude E6400 with 2.53GHz and 3.48G memory. We test the proposed algorithm on both static unimodal road networks and dynamic multimodal networks with random selected OD pairs. As the datasets provide consistent x-y coordinates of nodes, we can correctly test the proposed algorithm and compare its performance with CPLEX optimization solver and with labeled-Dijkstra’s shortest path algorithm. For each instances, each link is associated with two different resource consumptions based on uniform distribution within \([0, 1]\). As for the random query settings, two random resource bounds are used with randomly selected source to a fixed target. The numerical study on static networks is shown in Table 2. We run 100 queries and obtain its average performance. It indicates the proposed A*-label algorithm found the solution much faster than CPLEX solver. The speedup ratio is about \( 10^2 \) compared to CPLEX solver. The tests on classical labeled-Dijkstra’s algorithm show it fail to find solutions for most queries of all tested networks.

For dynamic multimodal shortest path problem with multiple constraints, as the selected test networks contain only road network, we need to generate a set of public transportation networks and connect them with the road network. Two public modes (metro and tram) are generated, each with 10 lines running on two directions in which 10 to 20 stops are generated by random walk method on the road network. We use piecewise linear function with two peak-hour periods for road link travel time computation. Metro and tramway operate on both directions with fixed timetables. The random query setting is similar with static case with additional random departure time setting. The computational result is shown in Table 2. The proposed method can find all solutions for instance 1 and 2. However, it cannot find all solutions for large cases (instance 3 and 4), which need to be further studied by combing other pruning strategies to accelerate the searching speed and reduce memory consumption in the searching process.

| Instance            | \(|V|\) | \(|E|\) | Static case | Dynamic case |
|---------------------|--------|--------|-------------|--------------|
|                     |        |        | CPLEX (1)   | A*-label (2) | Speed-up ratio (1)/(2) | A*-label |
|                     | Time   | Time   | Time        | Time         | Time |
| Lorraine (subregion 1) | 1459   | 4048   | 4.638       | 0.034        | 138  | 0.215 |
| Lorraine (subregion 2) | 3364   | 9262   | 11.447      | 0.191        | 60   | 0.672 |
| Lorraine (subregion 3) | 6462   | 17596  | 6.108       | 0.031        | 195  | 1.266 |
| Lorraine (all)       | 8267   | 22308  | 23.578      | 1.121        | 21   | 4.789 |

Remark: 1. Time (by second) is the time per query, executed on Dell Latitude E6400 with 2.53GHz and 3.48G memory, based on the average query time of obtained solutions
2. Resource bounds are set randomly between [10, 40] for subregions 1 and 2, and between [30, 60] for subregion 3 and the whole region, respectively.
3. For large cases (subregion 3 and the whole region), the A*-label algorithm cannot find all solutions of the queries due to computer memory limits. The average query time above is based on solved queries.

5. Conclusion

In this study, we present an A*-label setting algorithm for computing multimodal constrained shortest path for realistic network. The proposed method can efficiently solve static / dynamic multimodal shortest path problems with multiple constraints. By incorporating the A* algorithm and Access Node routing technique, the travel time lower bound from any node of the multilevel network to the target can be efficiently evaluated. The computational study on 4 realistic instances shows the proposed method obtains the same solution as 0-1 integer programming solver with a speed-up ratio of $10^2$. Further extensions include the combination of other speed-up techniques for large scale cases with multiple resource constraints and its application on multimodal traveler’s route advisory guidance system. The application of the proposed method on mobility analysis in cross border areas of Luxembourg city is currently under study.

Acknowledgements

The author thanks sincerely the support of Fonds National de la Recherche Luxembourg for the ACROSS project (Assessing the Socio-Cultural Effects on Mobility Behaviours in Cross-Border Areas), CORE 2010 (SR/783207), and Mr. Marc Schneider, CEPS/INSTEAD, for his kindly providing the road network of Lorraine region, France.

References

Aneja Y.P., Aggarwal V, & Nair K.P.K. (1983). Shortest chain subject to side constraints. Networks, 13, 295–302.
Bar-Gera, H. (2011). Transportation network test problems. http://www.bgu.ac.il/~bargera/tnp/
Barrett, C, Bisset, K, Holzer, M., Konjevod, G., Marathe, M., & Wagner D (2008). Engineering Label-Constrained Shortest-Path Algorithms. Fleischer R, Xu J. eds. LNCS, 5034, pp.27–37. Springer.
Barrett, C., Jacob R., & Marathe, M. (2000). Formal-language constrained path problems. SIAM Journal of Computing, 30(3), 809–837.
Beasley, J., & Christofides, N. (1989). An Algorithm for the Resource Constrained Shortest Path Problem. Networks, 19, 379–394.
Festa, P. (2006). Shortest path algorithms. In: Handbook of optimization in telecommunications, pp.185-210. Springer.
Garey, M., & Johnson, D. (1979). Computer and intractability: a guide to the theory of NP-completeness. Freeman and company.
Handler, G., & Zang, I. (1980). A Dual Algorithm for the Constrained Shortest Path Problem. Networks, 10, 293-310.
Hart, P.E., Nilsson, N.J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on System Science and Cybernetics, SSC-4(2), 100-107.
Ikeda, T., Hsu, M.Y., Imai, H., Nishimura, S., Shimoura, H., Hashimoto, T., Tenmoku, K., & Mitoh, K. (1994). A Fast Algorithm for Finding Better Routes by A* Search Techniques. Proceeding of Vehicle Navigation and Information Systems Conference, IEEE, 291-296.
Lozano, A., & Storchi, G., (2001). Shortest viable path algorithm in multimodal networks. Transportation Research Part A, 35, 225-241.
Lozano, A., & Storchi, G., (2002). Shortest viable hyperpath in multimodal networks Transportation Research Part B, 36, 855-874.
Mendelso, A.O., & Wood, P.T., (1989). Finding regular simple paths in graph databases. Proceedings of the 15th international conference on very large data bases, 185-193.
Pajor, T., (2009). Multi-Modal Route Planning. PhD thesis, Universität Karlsruhe, Germany.
Sherali H.D., Hobeika A.G., & Kangwalklai S., (2003) Time-Dependent, Label-Constrained Shortest Path Problems with Applications. Transportation Science, 37(3):278–293.
Ziegelmann, M., (2001). Constrained shortest paths and related algorithms. PhD thesis, Universität des Saarlandes.