# Implementing Fastest Path Algorithms in a Decentralized Traffic Environment

Reiner Kriesten, UweKiencke

*Abstract*--**The numerous amount of existing Route Guidance Systems (RGS) leads to increasing efforts to integrate these stand-alone tools into an overall solution, possessing the ability to process information of all the individual systems. Especially in the fields of intermodal services and in order to combine RGS of neighboured regions enhanced developments can be regarded. As representatives of intermodal services, i.e. calculating the best ways of certain origin-demand matrices with respect to the simultaneous use of different Public Transportation Means (PTM) *and* Individual Transportation (IT), the European project *Marco Polo* [EK01] can be named as well as the German projects *Mobilist* [MOB02] or *Mobinet* [EK01], mainly trying to implement shortest path models under a star topology with distributed information storage. Also, Personal Digital Assistents (PDA) with integrated GPS modul are curently available [PTV 02], thus being able to perform intermodal navigation within the vehicle as well as by the use of PTM and for pedestrians.**

**Unfortunately, analysis of different path search algorithms is commonly done by comparing the amount of necessary instructions O(·) in possible net topologies. However, as computing power is in the meanwhile at a fairly high level, delay in a distributed environment can mainly be expected due to communication time. Dynamic calculations demand to transmit actual traffic conditions during several time periods, thus this paper examines the different routing strategies by evaluating the occuring message transmission time in common graph classes. It will be shown that possessing a star topology (one central server) Label-Setting algorithms can be proved to be superior in regard to Label-Correcting algorithms. In addition, considerable improvements will be achieved by parallel message transfer for possible next link investigations. Here, the paper proposes solutions with a profit in delays by a factor of $O'(n)$, where *n* denotes the number of nodes in a network.**

## I. Introduction

Specifying the fastest path in a net topology has yield to numerous solutions in literature, each of them possessing (dis-)advantages in certain graph classes. As these algorithms have to work in a fast way, the crucial criteria in analyzing the models is commonly determined by the complexity $O(\cdot)$ of necessary instructions in order to decrease required processor power.

However, increasing efforts to combine co-existing Route Guidance Systems (RGS) into single solution models exhibit concepts for adequate best path approaches in the resulting distributed environment. Here, significant delays during iteration result not mainly because of restrictions in computation power but rather in the fact of existing, inevitable communication between distributed computers and/or storage devices.

Especially in the field of intermodal services, i.e. calculating the best ways of certain origin-demand matrices with respect to the simultaneous use of different Public Transportation Means (PTM) *and* Individual Transportation (IT), calculations have to be based on various databases stored in (locally) distributed devices. As representative efforts in this field, the European project *Marco Polo* [EK01] can be named as well as the German projects *Mobilist* [MOB02] or *Mobinet* [EK01], mainly trying to implement shortest path models under a star topology with distributed information storage. Surely, the integration of several IT-Guidance Systems from locally distributed areas into one overall architecture also focuses on this challenge.

This paper investigates existing routing strategies in distributed environments by evaluating the occuring communication complexity $O'(\cdot)$ in order to perform the algorithm accurately, a consideration, which is fairly different from the commonly regarded amount of executed instructions $O'(\cdot)$ within the processor. Therefore, the beginning of the next section provides a short overview over the commonly used *Label-Setting (LS)* and *Label-Correcting (LC)* models and its implementation. Based on this knowledge, results will be worked out concerning on one side the total amount of necessary message transmission within a distributed environment, on the other side conclusions will be drawn how to efficiently set up parallel message transmission in order to additionally decrease computation time. It will be shown that in general LS algorithms are superior to LC realizations by a factor of O'(n), a fact which contradicts the requirement of a sorted storage of examined nodes in LS models (leading to and increased processor load but less communication delay).

Institute of Industrial Information Technology (IIIT), Hertzstr.16 Geb. 06.35, University of Karlsruhe (TH), Germany

Before analyzing message transmission complexity, the next section presents an overview over the relevant search algorithms.

## II. Fastest Path Algorithms

### A. The Basic Algorithm

The most relevant methods of Fastest Path algorithm's (FPA) within a given network network $G=(N,L)$, where $G$ denotes the set of nodes connected via links $L \subseteq N \times N$ possessing nonnegative impedances $c(l)$, can be classified into Label-Setting (LS) and Label-Correcting (LC) algorithms. The representation of mathematical formulation will be given with the Min-Plus Algebra,

$$\mathbb{R}_{\min} = \mathbb{R} \cup \{\infty\} \qquad (1)$$

which possess the characteristics of a monoid

$$(\mathbb{R}_{\min}, \oplus', \otimes) \qquad (2)$$

with additive neutral element $\infty$ using the compositions

$$a \oplus' b = \min\{a,b\}, \ a,b \in \mathbb{R}_{\min}, \qquad (3)$$

$$a \otimes b = a+b, \ a,b \in \mathbb{R}_{\min}. \qquad (4)$$

As path search algorithms mainly rely on comparisons of temporary specified paths (usage of $\oplus'$) and additions of segment impedances (usage of $\otimes$), the Min-Plus representation describes a convenient tool for evaluation. For detailed information it shall be refered to [BAC92], [KIE97].

The determination of a shortest way between the origin node $o \in N$ and a destination $d \in N$ results in a consecutive search in forward direction beginning from origin $o$. In each iteration of the algorithm, a node $k$ from the set $Q$ of so far reached nodes will be elected. Then, each successor $l \in S(k)$, i.e. nodes $l$ possessing a link $(k,l) \in L$ from $k$, will be analyzed in order to evaluate possible better connections from $o$ to $l$ via $k$ (if the node $l$ is not reached so far, surely the new connection displays the optimal one). Once the set $Q$ of investigated points is getting empty, the algorithm has found the optimal path $o \to d$ [CHR75]. The basic implementation of LS/LS algorithms are as follows:

***Basic Implementation for LS/LC FPA's:***

Initialization: The initial distance between nodes $o,k \in N$ will be defined by

$$d(o,k) = \infty, \ o \neq k, \qquad (5)$$

$$d(o,o) = 0. \qquad (6)$$

The set $Q$ of initially reached nodes is declared by

$$Q = \{o\}. \qquad (7)$$

Furthermore let $p(k)$ denote the direct predecessor of $k$ under the current specified connection. So at the beginning we find

$$p(k) = \varnothing, \ k \neq o, \qquad (8)$$

$$p(o) = o. \qquad (9)$$

Iteration: For $Q \neq \varnothing$, perform the following instructions:

Choose $k \in Q$ by the logical rules of the elected LC/LS algorithm and set $Q = Q - \{k\}$.

Verify for all successors $l \in S(k)$ of node $k$ the condition of an eventual new ideal route:

$$d(i,k) \otimes c((k,l)) < d(i,l). \qquad (10)$$

If condition (10) holds, declare $p(l) = k$, $Q = Q \cup \{l\}$ and set the new distance of node $l$ to

$$d(i,l) = d(i,k) \otimes c((k,l)). \qquad (11)$$

Within step 0 of the algorithm, the choice of node $k \in Q$ depicts the crucial difference of LS/LC models. As the name *Label-Setting* indicates, these methods insert each (reachable) node exactly one time into the set $Q$. So once we find $k \in Q$, there is no better alternative route $o \to k$ available than the specified way, the label $d(o,k)$ remains unchanged after its first assignment.

Contrary, LC algorithms may insert a certain point $k$ several times into $Q$. Unfortunately, according to the Belman Principle all further paths $o \to l$ with in-between node $k$ have to be updated once again, as the new impedance $d_2(o,k)$ leads to the new cost value

$$d_2(o,l) = d_2(o,k) \otimes d(k,l) \qquad (12)$$

of route $o \to l$. However, LS models depend on an ordered storage of the set $Q$ because it has to be ensured that the ideal route to $k$ is already fixed when assigning $k \in Q$. The following subsections will give a short outline of variants in LS/LC implementations.

### B. Label-Correcting Algorithms

Label-Correcting algorithms are not restricted to one single inclusion of individual nodes $k \in N$ into $Q$, the selection criterias in step 0 are widely spread. The most common methods are listed below

1. Ford Algorithm: This approach is scanning the set $Q$ according to paths with smallest path length, hence representing a Breadth-First Search. In each iteration, one node possessing the least segments on its temporary route (independent from the impedance $d$) is chosen for further evaluation. Thus, the storage of $Q$ can be realized in a queue obeying the FIFO principle because new incoming nodes have per se an increased path length.

2. The LIFO Principle: Is $Q$ being realized as a stack, then the actual path keeps being evaluated until no further successor is achievable. These kinds of models are known as Depth-First Search and prefer routes with a high amount of segments.
3. Combined Strategies: Strategies combining Breadth-First and Depth-First characteristics have become more and more popular within the last years. As most important variants the Threshold Algorithms, the Dequeue Algorithm and the d'Esopo Algorithm can be named [DOM95] [SCO97].

### C. *Label-Setting Algorithms*

Including each node at most one time into $Q$ exhibits the selection of a node $k$ with minimal route costs $d(o,k)$ out of the set $Q$. Otherwise an eventually better way leading to $k$ can be found by an in-between point $l \in Q$ possessing less impedance,

$$d(o,l) \otimes d(l,k) < d(o,k). \qquad (13)$$

As the sequence of analyzing $Q$ is completely determined, all realizations of LS algorithms are commonly refererred as Dijkstra Algorithm. Differences exist in the way how the set $Q$ is sorted in order to access the smallest route impedances.

1. Unsorted Dijkstra Algorithm: the set $Q$ is not being ordered in any iteration. Identifying the minimum $\oplus'_{\{k \in Q\}} d(o,k)$ has to be accomplished by comparing all nodes $k \in Q$.
2. Completely sorted Dijkstra Algorithm: at each instance the set $Q$ is totally sorted by its route costs. The ordering can be performed by one of the various sorting algorithms [REM99].
3. Heap-sorted Dijkstra Algorithms: Heaps are defined as partly ordered arrays $H[1..n]$, characterized by the conditions $H[i] \le H[2i]$ and $H[i] \le H[2i+1]$. Due to the party ordered storage, savings during the determination of extremas can be obtained.
4. Bucket-sorted Dijkstra Algorithms: Possible route impedances are divided into several intervals in which specified paths are categorized.

The following table presents an overview over the number of instructions $O(\cdot)$ which have to be performed using the different implementations [DOM95]. Here, $n = |N|$ displays the number of nodes in a network, $m = |L|$ depicts the amount of links and $c_{\max}$ is an upper bound for possible segment costs,

$$\forall l \in L: \; c(l) \le c_{\max}. \qquad (14)$$

| **Algorithm** | **Complexity** $\mathbf{O}(\cdot)$ |
|---|---|
| Ford Algorithm | $O(nm)$ |
| D'Esopo Algorithm | $O(n2^n)$ |
| Dequeue Algorithm | $O(n^2 m)$ |
| Unsorted Dijkstra Algorithm | $O(n^2)$ |
| Heap-sorted Dijkstra Algorithm (depending on realization) | $O(m + nc_{\max})$, $O(m \log_d n)$, $O(m + n \log_d n)$. |
| Bucket-sorted Dijkstra Algorithm | $O(m + nc_{\max})$ |
| Completely sorted Dijkstra Algorithm | $O(n^2 \log_d n)$ |

Table 1: Complexity $O(\cdot)$ for FPA's (#instructions)

## III. Adaption of FPA's to distributed environments

As the integration of intermodal and/or locally distributed RGS leads to decentralized networks, the question how to realize FPA's within this distributed environment arises. The following categorization of net topologies can be given:
RGS without essential delay due to message transmission are defined as centralized systems [SUN96]. Here, only the number of executed instructions are relevant for efficiency. Computers and/or Information Storage Devices (ISD) are connected to one central instance (CI). This setup leads to a star topology according to figure 1(a), where all information is transmitted to the main computer, being solely responsible for the execution of instructions.
The decentralized devices can be individually interconnected without central instance, see figure 1(b). In this case no superior node exists, so problems arise in the adjustment and provision of actual dynamic information. Several devices can be combined to a network via star topology and/or flat hierarchy. Integrating some of these networks into one single architecture, topologies as depicted in figure 1(c) are possible.
In a decentralized network the far most popular way to setup the topology is realized by a star network. This is due to the fact that each node only needs to transmit relevant information to the main node and not performing 1:n communication. In addition, just one server is responsible for the actuality and distribution of information, so adjustments between nodes (concerning e.g. current specified paths, costs, etc.) don't have to be taken into consideration. Taking this into account the analysis of message transmission complexity is further regarded under the assumption of a star topology with distributed information (instructions are executed by the control instance).

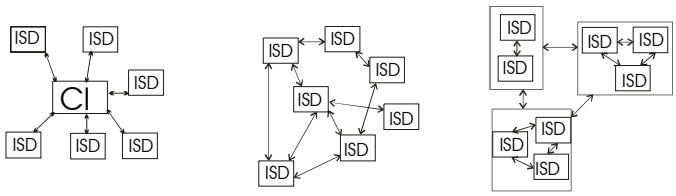


Figure 1: Topologies in a decentralized environment

The linkage of routing systems via a star topology connects each ISD/computer directly to a main node, following denoted by CI (central instance). All essential information will be transferred to the CI, consequently instructions (relying on the information) should run at this node. As soon as the algorithm has to access certain knowledge such as dynamic varying segment impedances $c(l)$, message transmission toward the CI is inevitable. In this context, data transfer between individual ISD's is not intended, hence complexity evaluation is based on communication with the server.

Clearly, allowing parallel messaging leads to savings in time. Therefore, beside the number of overall transfers the latter definition of required time units is of main interest:

***Definition 1**: Let $O_c'(\cdot)$ denote the amount of overall transmitted messages between adjacent nodes in an existing network $G=(N,L)$.*

***Definition 2**: Let the required time for a transmitting a message between adjacent nodes be 1 time unit. Then $O_t'(\cdot)$ depicts the complexity of time units an algorithm needs for message transfer.*

Before a deeper evaluation of the different LC/LS methods, a principle consideration is helpful under what circumstances communication occurs. Assuming a star topology the basic algorithm for FPA's given in the last section can be implemented as follows:

the storage of the set $Q$ and the variables $d(o,k)$, $k=1,...,n$, is recommended in the CI, as execution is performed here.

the initialization does not lead to any communication as it commonly holds that $d(i,k)=\infty$, $k\neq o$.

the selection of $k\in Q$ takes place at the main instance.

Only the knowledge of logical criterias is decisive for correct selection, no data transfer is demanded. So concerning iteration step 0, the dynamic varying link costs $c(l)$ for $l\in S(k)$ are sufficient for transmission. Based on that information, the comparison

$$d(i,k)\otimes c((k,l))<d(i,l) \qquad (15)$$

can be realized as well as the further specified assignments.

Complexity Analysis $O_c(\cdot),O_t(\cdot)$ of FPA Message Transmission

During each iteration step the data $\{c(k,l)\mid l\in S(k)\}$ has to be sent to the server, so the overall message amount depends significantly on the amount of fulfilled iteration cycles and therefore on the logical sequence of choosing $k\in Q$. If $i$ denotes the number of executed iteration steps, it follows directly that

$$O_c'(\cdot)=O_c'(i\mid S(k)\mid)\leq O_c'(in) \quad (16)$$

Allowing parallel data messaging, this implementation claims only 1 time unit sending $c(k,l)$ for *all* nodes $l\in S(k)$ in contrast to $\mid S(k)\mid\leq n$ units for serial transmission. Thus, gainings can by achieved by a factor of $\mid S(k)\mid\leq n$ leading directly to

$$O_t'(\cdot)=O_t'(i). \quad (17)$$

Specifying boundaries for individual LC/LS algorithms requires results about the maximum number of possible iterations. In the case of LS methods conclusions can be easily drawn.

message Evaluation for LS algorithms

Differences in the realization of LS algorithms can be regarded concerning the storage of the set $Q$ and the way to extract its minimum route costs. As already explained, this operation doesn't have any influence on necessary data transfer, hence the following results are valid for all LS variants.

Fortunately, we find the maximum number of iterations to be bordered by $n=\mid N\mid$ as each point will be examined maximal one time. So upper boundaries can be denounced by

$$O_c'(\cdot)=O_c'(n\mid S(k)\mid)\leq O_c'(n^2), \qquad (18)$$

$$O_t'(\cdot)=O_t'(n). \quad (19)$$

message Evaluation for LC algorithms

In the case of LC methods, the number of iteration cycles depends in a significant way on the given graph structure and the logic of choosing nodes $k\in Q$, a topic being widely researched [CHE93] However, a strong influence of communication efforts in regard to execution time should lead to efforts minimizing the amount of iteration steps. The following theorem provides an upper bound for the most popular Label-Correcting method, the Ford Algorithm.

*Theorem 1: In a given network $G=(N,L)$, each node is chosen at most $\mid V\mid -1=n-1$ times in $Q$ performing a Breadth First Search. Furthermore, the communication complexity can be bounded by the upper border*

$$O_c'(\cdot)=O_c'(n^3), \qquad (20)$$

$$O_t'(\cdot)=O_t'(n^2). \quad (21)$$

*Proof:* The proof of the theorem is mainly relying on the next lemma:

*Lemma 1: Having specified a path $\Gamma$ with path length $\mid\Gamma\mid=n-1$ while executing the Ford Algorithm leads to the fact that $\Gamma$ is not inserted once again into $Q$. To be more exact, the leaf $k$ of the path is remaining the*

*end of the path as long as* $\Gamma$ *depicts the shortest way* $o \to k$.

*throughout the whole algorithm, not being reinserted into* $Q$.

*Proof (Lemma)*: A further node in $\Gamma$ leads to a loop as $|N| = n$. Let $l \in N$ denote a point being two times an element of a path. The second stop in $l$ is determined by the fact

$$d(o,p) \otimes c(p,l) < d(o,l), \qquad (22)$$

where $p$ depicts the direct predecessor of $l$ before the second stop. As node $l$ is lying on the way $o \to p$, we find

$$d(o,p) \geq d(o,l), \qquad (23)$$

resulting in a contradiction to the upper equation.

*q.e.d (Lemma)*

Proceeding with the proof the theorem, $k$ denotes again the leaf of a current path $\Gamma$ with $|\Gamma| = p$. Deleting $k$ out of the set $Q$ requires the specification of a path to $g \in S(k)$ with $|o \to k \to g| = p+1$. By reinserting node $k$ into $Q$, Breadth-First Search exhibits $k$ to be successor of an element $l$, $k \in S(l)$, with path length

$$|o \to l| \geq p \qquad (24)$$

(otherwise the route $o \to l$ would have been taken into account before $o \to k$). Proposition 1 gives an upper bound for specified routes $|\Gamma| \leq n-1$, thus each node is inserted at most $n-1$ times into $Q$. With equation (16) it follows directly

$$O_c'(\cdot) = O_c'(n^3) \quad (25)$$

as each of the $n$ nodes is chosen at most $n-1$ times. Parallel transmission reduces required time at each iteration by the factor $|S(k)| \leq n$, hence

$$O_t'(\cdot) = O_t'(n^2). \quad (26) \qquad \textit{q.e.d.}$$

The following table summarizes the communication complexities $O_c'(\cdot)$, $O_t'(\cdot)$ for LS/LC algorithms under the assumed star topology.

| **Algorithm** | $\mathbf{O_c'(\cdot)}$ | $\mathbf{O_t'(\cdot)}$ |
|---|---|---|
| LS Algorithms (all realizations) | $O_c'(n^2)$ | $O_t'(n)$ |
| Ford Algorithm | $O_c'(n^3)$ | $O_t'(n^2)$ |

Table 2: Comunication Complexity $O'(\cdot)$ for FPA's

## IV. Conclusion

This paper investigates the setup of routing strategies in distributed environments, an essential topic for combining co-existing Route Guidance Systems into one single solution. Here, delays due to necessary communication is getting more and more significant in comparison to execution time, so the amount of transmitted messages is playing an important factor for efficient implementation beside the number of instructions to be performed. After specifying possible ways to setup networks of combined RGS, this paper evaluates the communication load for the most popular FPA's. Analysis is done by determing the total amount of messages as well as time savings due to parallel data transmission. In this context it is proved that assuming a star topology (one central server), Label-Setting algorithms are superior to Label-Correcting algorithms by a factor of $O'(n)$ concerning the overall number of messages as well as parallel messaging. As processor power is getting increasingly more powerful, these results give helpful indications to efficiently combine autonomous RGS for fastest path determination.

## V. References


[1] [BAC92] Baccelli, F., Cohen, G. (1992): Synchronization and Linearity. An Algebra for Discrete Event Systems. Wiley & Son, New York.

[2] [CHE93] Cherkassky, B., Goldberg, A., Radzik, K. (1993): Shortest Paths Algorithms: Theory and Experimental Evaluation. ACM-SIAM Symposium on Discrete Algorithms (SODA), Texas.

[3] [CHR75] Christofides, N. (1975): Graph Theory: An Algorithmic Approach. Academic Press, San Diego.

[4] [DOM95] Domschke, W. (1995): Logistik: Transport. Oldenbourg Verlag München (Germany).

[5] [EK01] Europäische Kommission (2001): Das Programm Marco Polo. Konsultationsdokument Europäische Kommission. Generaldirektion Energie und Verkehr, Direktion Landverkehr, Brüssel.

[6] [HAL01]: Halbritter, G. et al (2001): Verkehr in Ballungsräumen. Optionen für eine effizientere und umweltverträglichere Gestaltung. Institut für Technikfolgenabschätzung und Systemanalyse, Forschungszentrum Karlsruhe (Germany).

[7] [KIE97] Kiencke, U. (1997): Ereignisdiskrete Systeme. Oldenbourg Verlag, München (Germany).

[8] [MOB02]: Stadt Stuttgart (2002): Mobilist. Mobilität im Ballungsraum Stuttgart. http://www.mobilist.de.

[9] [PTV 02]: PTV AG (2002): ptv NaviGuide – Innovative Offboard Navigation. http://www.ptv.de/cgi-bin/produkte/naviguide.pl.

[10] [REM99] Rembold. U. (1999): Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Carl Hanser Verlag, Munich (Germany).

[11] [SCO97] Scott, K. et al. (1997): Finding alternatives to the best path. Proceedings of the 76th annual meeting of The Transportation Research Board, paper number 970682, Washington.

[12] [SUN96] Sun, W. (1996): Optimale Steuerung verteilter, ereignisdiskreter Systeme. Ph.D. thesis, University of Karlsruhe (Germany).