# A Survey on 3D Position-based Routing Algorithms for DANETs

Claudio Enrico Palazzi, *Member, IEEE,* Armir Bujari, and Daniele Ronzani

***Abstract*—The abstract goes here.**

***Index Terms*—Wireless Communications, Routing 3D, Algorithm, MANET, Mobility, Forwarding.**

## I. Introduction

*Mobile ad-hoc Networks* (MANTETs) are autonomous, distributed and self-configuring networks in which mobile terminals are connected by a wireless connection, even through multi-hop communications. This type of network operates without a fixed infrastructure or centralized administration and its flexibility makes it suitable to a wide set of scenarios (e.g., rescue teams, disaster or dangerous environments, military areas, underwater networking). The main characteristic of a MANET is the frequent change of its topology, due to the motion of the terminals. This makes routing discovery and maintenance a very challenging task.

Evolutions of MANET have considered vehicles (VANETs - Vehicular Ad-hoc Networks), sensors (WSN - Wireless Sensor Networks) and, more recently, microaerial vehicles (drones or UAVs), which can create a *Drone Ad-hoc Network* (DANET). These network subclasses differ from MANET for several aspects, such as energy constraint, degree of mobility, type of devices and communication. A DANET generalizes the topology from a 2D topology to a 3D one with a free movement scheme, due to the capability of drones to fly, also in autonomous mode. This trend is continuing to attract researches and practitioners, as well as to become increasingly present in real application. Furthermore, this is a very interesting and challenging scenario, especially when considering the routing process.

In a generic MANET, the wireless nature of communication channel, the mobility of the nodes, and the lack of a communication infrastructure, make the problem of routing very complex. Mobile hosts move free in the space, resulting in a dynamic network with potentially rapid changes of its topology, without some notifications. Moreover, the mobile hosts often use batteries which have a limited energy supply, that brings to an energy comsumption problem in routing. *Position-based* routing protocols (or geographic-based protocols) try to tackle these issues, using a different mechanism for path discovery. These routing protocols use the geographic position of the nodes to perform packet forwarding, exploiting the recent availability of small, inexpensive, low-power *Global Position System* (GPS) receivers. Each node determines it own geographic position using GPS, and bases the routing decision only on the position of itself, the destination's position and the neighbors' position. This local knowledge of the network make position-based routing protocols more scalable towards large and high mobile networks, since the nodes do not have to explore the status of the whole network and ever have to store routing tables nor do they need to transmit control messages in the entire network. This work focuses on the analysis of the state-of-the-art of 3D position-based routing protocols, and on a performance comparison in a same simulation scenario, that represents a 3D topology network.

Name
name2

The rest of the paper is organized as follows. Section 2 introduces the routing in MANETs, providing a complete taxonomy of the routing protocols, with particular reference to the position-based protocols. Section 3 describes the performance metrics used for the performance comparison. Section 4 explore the state-of-the-art of of position-based protcols, with an accurate description of the algorithm functioning in each protocol. Section 5 describes the simulation environment and results of the performance evaluation. Finally, section 6 concludes the paper.

## II. Routing in Mobile Ad-hoc Networks

A MANET consists of a set of mobile nodes that communicate with each other over wireless links, capable of operating without centralized control or established infratructure. In this type of networks a *multi-hop* routing is needed, since a node needing to send a packet to another node can be not communicate directly with it. There are several situations in which MANETs are desiderable, e.g., in scenarios where infrastructure is not feasible or cost-effective (environmental monitoring, disaster relief, tactical area, etc.).

Recently, with the advent of new technologies that make it possible to miniaturize complex electronic systems, different gadget able to move and fly autonomously o remotely controlled, were born; these are called *unmanned aircraft vehicles* (UAVs), or, more simply, drones. The use of multiple UAVs is playing an important role, in particular considering the possibility to create communications between them, resulting in a DANET. This type of network can be used in a growing number of civil applications, such as policing, firefighting and nonmilitary security work, but often preferred in tactical, battlefield and dangerous scenarios operated without a human presence.

Since MANETs are characterized by frequently topology changes, energy comsumption problems and limited bandwidth, there are various challenges, in particular in routing process. So, the utility of routing mechanism makes it an integral part of any network and there exists a multitude of routing protocols. Nowadays, there are many routing protocols proposed for MANETs to address multi-hop routing. In general there protocols can be categorized in topology-based and position-based approaches. Topology-based routing protocols use the information about links to route the packets, position-based ones use the location information of nodes to make a packet forwarding decision.

Topology-based protocols consider the network topology and employ *routing tables*, that specify the path or the next-hop to route a packet from a sender to a recipient. There are three strategy that use topology information: proactive, reactive and hybrid. A protocol is said *proactive* when each node keeps an up-to-date information reflecting the state of the network and this information is used when a message should be sent. A protocol is said *reactive* when the routing patch is created only when necessary and it is preferred in mobile networks. A significative example of reactive protocol is named *AODV* (*Ad-hoc On Demand Distance Vector*). *Hybrid* protocols combine the advantages of proactive and reactive protocols.

Due to change in topology of mobile environment, the maintenance of routing tables require an excessive overhead for information updates (continuosly exchange of control messages). Furthermore, in large networks this situation can only get worse, and routing tables size become very high. For this reason, topology-based protocols become almost useless in more dynamic and large networks. Position-based approaches are introduced to eliminate some of these limitations MANETs: they do not need to establish and maintains routes, thereby elimintating routing table construction and maintenance. This aspects make position-based protocols more scalable, unlike topology-based ones.

## A. Position-based Routing Protocols

Position-based (or geographic) routing protocols use the position of the nodes in the network to make forwarding decisions. The first proposed protocols that use geographic information were intended to act as support to the topology-based protocols, to limit the propagation of route request packets into a restricted area. For instance, one of the early proposed protocol was *Location Aided Routing* (LAR), that generally limits the propagation area of packets into a rectangle containing source node and destination node positions. Position-based routing protocols are local, because a node forwards a message based only on its position, the position of the destination and the position of its neighbors to which it can communicate directly. Therefore, these protocols do not require a global knowledge of the network, but they rely on having only a piece of network information. To obtain coordinates, nodes use a *location service* such as GPS (*Global Positionin System*), or other types of location services. To acquire the position of the neighbors, nodes are provided of a beaconing mechanism in which each node sends a beacon to its neighbor nodes, containing its position. *I*n general, position-based protocols have the following characteristics:

- Each node can determine its position (longitude, latitude and altitude), the position of its neighbors (usually 1 hop) and the position of the destination.
- Nodes need to store the information about its neighbors (position, speed, direction) in a *neighbors table*.
- The next-hop decision can be made based on the location of the current node (the node that holds the packet), its neighbor nodes and destination node.

## B. Taxonomy

In the years there was make several taxonomies to classificate position-based routing protocols. In this survey we classificate these according to two types of strategies: path strategy and forwarding strategy.

*1) Path Strategy:* represents how a packet traverses a network. The packet can uses either a single path to reach the destination, or multiple paths. Single path strategy requires that each node forwards the packet to only one of its neighbors. So, there is one copy of a packet in the whole network. Algorithms that uses single path strategies may be even more robust and with less communication overhead. Multipath strategy allows a node to forward multiple copies of the same packet to several neighbors. This strategy is also defined as *flooding*. Flooding can be costly in terms of wasted bandwidth, because packet become duplicated in the network. Moreover, duplicate packets may circulate forever in loop, unless certain precaution.

*2) Forwarding Strategy:* describes the forwarding criteria used to send a packet from the current node to a neighbor node in a single step, and it regards especially single path strategies. Every strategy exploits different methods and geometric models. The three main forwarding strategies are *deterministic progress-based*, *randomized progress-based* and *face-based*. In deterministic progress-based routing algorithms, the current node (the node holding the packet) forwards the packet at every step to one of its neighbors that make progress to the destination. Randomized progress-based strategy is similar to deterministic progress-based method but in this case the next node is chosen uniformly at random or according to a probability distribution, from the set of neighbor nodes or candidate nodes. Face-based is a strategy that allows to arrive at a delivery-rate close to 100% in some cases. In the context of a two-dimensional space, face-based algorithms allows progress between the faces defined by the nodes considering the right-hand rule, with which always guarantee reaching the destination. In the three-dimensional space, the concept of "*face*" can not be extended, but some approaches have been proposed that are based mainly on the projection of points in a plane.

## C. Routing in 3D Networks

For 2D networks, a lot of research has been done about the problem of position-based routing; however, in real scenarios, nodes may be distributed in 3D space and the extension of 2D routing protocols into 3D protocols is not trivial. Topology-based routing protocols are not sensitive to the addition of the third dimension because they rely on a link-state system knowledge. Instead, position-based protocols are based on the spatial position. In a three-dimensional space some assumptions made in 2D, such as the ability to extract planar

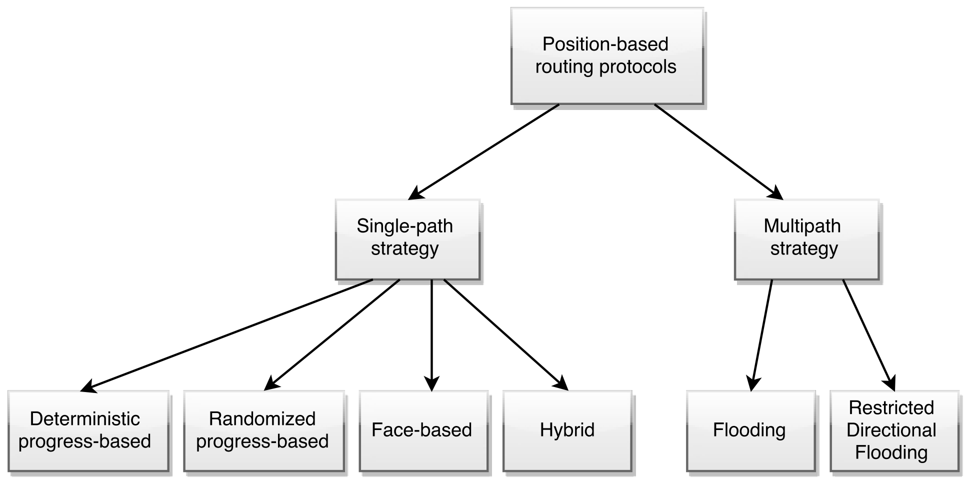

Position-based routing protocols
Single-path strategy
Multipath strategy
Deterministic progress-based
Randomized progress-based
Face-based
Hybrid
Flooding
Restricted Directional Flooding

Fig. 1. Taxonomy of 3D routing protocols.

subgraphs, break down. Durocher *et al.* [**?**] shows the impossibility of routing protocols that guarantee delivery in three-dimensional ad-hoc networks, when nodes are constrained to have information only about their $k$-hop neighborhood, in contrast to the two-dimensional case, where a protocol that uses 1-local algorithm, such as face routing (see section **??**), guarantees delivery. This leads the problem of finding other solutions that can guarantee the delivery of packets, with the least use of resources. Some works have proposed several algorithms that try to achieve a higher delivery rate, with a smaller number of node involved. This survey takes these and compares them in a fits-all simulation, to obtaining their performances.

### D. Notation and model

In the following we will use the seguent conventions and notations. A model of MANET is represented, in R$^2$ and R$^3$ spaces, by a geometric graph $G = (V, E)$, consisting of a finite set $V = v_1, v_2, ..., v_N$ of nodes and a subset $E$ of cartesian product $V \times V$, the elements called edges (links from a node to another). All nodes have the same communication range $r$, which is represented as a sphere in 3D space. The graphs thus obtained is called *Unit Ball Graph*, $UBG(V, r)$. We define $dist(u, v)$ as the distance between two nodes $u$ and $v$, given by the formula of the Euclidean distance:

$$dist(u, v) = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2}.$$

Two nodes are said to be neighboring and connected by a link if the Euclidean distance is at most $r$, and, for a node $u$, we define the set of its neighbors as $N(c)$. A path from a node $s$ to a node $d$ is a sequence of nodes $s = v_1, v_2, ..., v_k = d$, such that $v_i$ and $v_{i+1}$, $1 \leq k - 1$, are neighbors.

In order to provide a uniform and fair treatment of all algorithms, a common terminology to define uniform concepts is introduced.

- The *source node* is the node that sends the packet, and *destination node* is the node that receives the packet. They are called respectively $s$ and $d$.
- The *current node* is the node that applies the algorithm at a given time and that has in its memory the package to be forwarded, and is called $c$.
- The *previous node* is the node that sent the packet to $c$ in the previous step, and is called $prev$.
- The *neighborhood* of the current node $c$ is the set of all nodes connected directly to $c$, called $N(c)$. The size of $N(c)$ is indicated with $n$.
- The sphere centered at node $c$ with radius $r$ is called $ball(u, r)$ and covers the transmission area of $c$ in three-dimensional space.

## III. Performance Metrics

In this section, we introduce the metrics of interest to evaluate the performances of the routing protocols described in this survey.

### A. Delivery Rate

*Delivery rate* is the ratio of the number of packets received by the recipient (or recipients) to the number of packets sent by the source (or sources). The primary objective of a routing algorithm is to guarantee the delivery of all the packets, that this means a 100% delivery rate.

### B. Path Dilation

*Path dilation* or *Stretch factor* is the ratio of the number of hops traversed by the packet to the number of hops of the minimum path.

## IV. State of the Art

In this section, we discuss the state-of-the-art of the position-based routing protocols in 3D MANETs. In particular, we describe the forwarding algorithms employed by the routing protocols.

The first four are simply an extension of their 2D counterparts, as we need only the definifion of Euclidean distance between two points $u$ and $v$ in three dimesions and the definition of the of the sphere with center $c$ and radius $r$.

### A. Greedy

*Greedy* is a simple progress-based forwarding strategy. A node forwards the packet to the neighbor node that minimizes the distance to the destination node. With *Greedy*, the current node's distance is not compared against the distances of its neighbors, but it is used to select the forward nodes and the backward nodes: if there is no neighbor node closer to the destination node than the current one, the algorithm fails. More precisely, there are two forwarding algorithms that use greedy strategy, and differ on which neighbors consider on the choice:

- *Greedy* [2]: the current node $c$ forwards the packet to one of its neighbors that is closer to $d$ than $c$ and any other neighbors.
- *GEDIR* [3]: the current node $c$ forwards the packet to one of its neighbors that is closer to $d$ than any other neighbors, but not necessarily closer to $d$ than node $c$ itself.

In *GEDIR*, all neighbors are considered. So, also the nodes that are in backward direction can be chosen and the only kind of loop that may be formed using this algorithm is the local loop between $c$ and the node $prev$ that sent the message to $c$ in the previous step [3], but this case is avoided by the assumption mentioned follow. If the next node for forwarding is the node that sent message to $c$, it means that the packet has reached a local minimum $c$ and then the algorithm fails. Instead, in *Greedy*, only the neighbors that are closer to the destination than $c$ are considered. If no one is closer to $d$, algorithm fails. Fig. 2 shows an example of choosing of next node using progress-based algorithms. The current node $c$ has five nodes that are in the direction of destination node $d$ inside its range, but with *Greedy* the node that is closer to the destination is 2 (the closer to the dotted arc). In this survey we refer to *Greedy* algorithm and specify when we consider *GEDIR* (for example in *PAB3D*).

### B. Compass

*Compass* (or *Directional, DIR*) algorithm [4] uses the direction of nodes to selects the best forwarding node. The current node $c$ uses the location information of $d$ to calculate its direction. Then, it forwards the packet to the neighbor node $u$ such that the direction $cu$ is closest to the direction $cd$, that is the neighbor node $u$ that minimizes the angle between $c$ and $d$. In Fig. 2 node $c$ chooses node 6 as the next node, since the angle $\angle 6cd$ is the smallest among all. *Compass* is not a loop-free algorithm.

### C. Most Forward

*Most Forward*, (*MFR - Most Forward Routing*) algorithm [5] is very similar to *Greedy*, but, in this case, the current node $c$ forwards the packet to the neighbor node $u$ whose projection on the line between $c$ and $d$ is closer to $d$. If the packet reaches a local minimum, the algorithm fails. In most cases *MFR* chooses the same path of *Greedy*. In Fig. 2 node $c$ selects node 5 as next node, since the latter has the smallest projected distance to $d$, on the line $sd$. *MFR* is a *loop-free* algorithm.

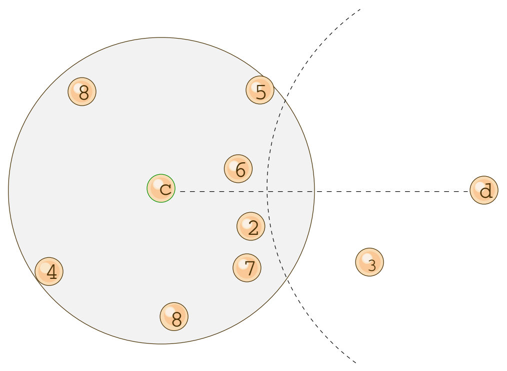


Fig. 2. Illustration of several next nodes chosen by $c$ using the progress-based forwarding strategies.

### D. Ellipsoid

In the *Ellipsoid* algorithm [6], the current node $c$ forwards the packet to the neighbor node $u$ that minimizes the sum of the distance from node $c$ to $u$ and the distance from node $u$ and the destination node $d$. Unfortunately, if the packet reaches a local minimum, the algorithm fails. In Fig. 2 the next node chosen is 6, using *Ellipsoid*.

### E. PAB3D

*PAB3D* is a randomized algorithm that tries to solve the local minimum problem described in previous section, by choosing the next node randomly from a subset of the current node's neighbors that makes progress to the destination. Already since 1984, some authors have tried to put the concepts of randomization algorithms in order to avoid the emergence of loops [7], [8], [9] using the concept of the random walk. Tipically, randomization performs as follow: let $cw(u)$ be the neighbor node in $N(c)$, over the line $cd$, that minimizes one of the progress values defined in previous progress-based algorithms (distance, angle, etc.) and let $ccw(u)$ be the neighbor node in $N(c)$, under the line $cd$, that minimizes the same one; then, randomized algorithms moves the packet to one of $\{cw(u), cww(u)\}$ with equal probability or a probability weighted according to the progress values. Routing algorithms that use randomization are usually considered to fail when the number of hops in the path computed so far exceeds a threshold value, here called $TTLR$ (*Time To Live Random*).
Extension of randomized algorithm in 3D is not trivial, because it is not obvious to choose the best way to determine candidate neighbors. The reason is that in a three-dimensional graph there is no concept of above and below a line passing from source to destination. Therefore, in [10] authors propose an extension for the randomized-based algorithm from 2D to 3D, that uses the concept of 3D planes. This new algorithm is called *AB3D* (*Above/Below 3D*) and, instead of a line, it uses a plane to divide the 3D space in two choosing regions. In this document we generalize the concept of randomized-based algorithm defining a parametric algorithm, named *PAB3D* (*Parametrized AB3D*), that has four attributes:

- $m$: are possible candidate neighbors to choose from $N(c)$.
- $R$: is the name of progress-based strategy used to choose the $m$ candidates, and it is one of $R$ (Random), $CM$

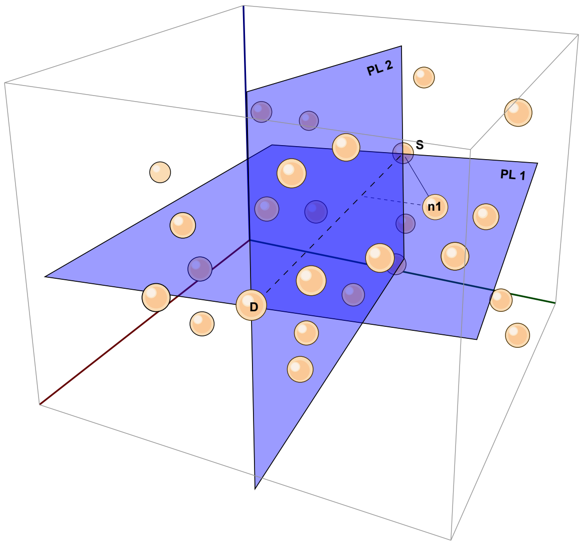


Fig. 3. In AB3D, plane $PL_1$ passes through $s$, $d$ and $n_1$, plane $PL_2$ is orthogonal to $PL_1$. Both planes contain the line $sd$.

(Compass), $GR$ (Greedy) or $MF$ (most Forward), as in *AB3D*.

- $S$: is used to represent the probability weighting when randomly choosing, and it is one of $U$ (Uniform), $D$ (Distance) , $A$ (Angle), $PD$ (Projection Distance). Probability weightings are defined as in *AB3D*.
- $ab$ (above-below): is a boolean flag indicating whether to define the candidates over and below the Plane $PL_1$ (or even over and below the plane $PL_2$, if $m = 5$), or select candidates without considering the planes.

The main difference from *AB3D* is the adding of parameter $ab$, that selects if we have to use the plane or not. The steps of the algorithm are the following. The current node $c$ selects a node $n_1$ from its neighbors, chosen according to the method defined in $R$ ($CM$, $GR$ or $MF$). With $ab = YES$, if $m$ is 3, we define the plane $PL_1$ identified by the three nodes $s$, $d$ and $n_1$. If $m$ is 5, define also the plane $PL_2$ that is perpendicular to $PL_1$ and passes through $c$ and $d$, such that the intersection line between the two planes is the line $cd$. Fig. 3 shows an example of network graph and its subdivision with the two planes. Then, if the parameter $m$ was 3, *PAB3D* select another two nodes, in addition to $n_1$. One neighbor $n_2$ is chosen from the above of the plane $PL_1$ according to $R$ and one neighbor $n_3$ is chosen from the below of the plane $PL_1$ according to $R$. If $m$ was 5, in addition to $n_1$, the algorithm choose from $N(c)$ four neighbors $n_2$, $n_3$, $n_4$, $n_5$ each one in one side of the four regions that result from the intersection between $PL_1$ and $PL_2$. Once these candidates are determined, node $c$ selects one of these nodes randomly, according to the probability weighting determined by $S$, and forwards the packet to the chosen node. Instead, with $ab = NO$, the $m$ candidates are chosen without a plane subdivision.

### F. Greedy-Random-Greedy

*Greedy-Random-Greedy* (*GRG*) algorithm belongs to *progress-based/randomized-based* class, and uses *Greedy* as the primary stage and a randomized algorithm, such as *PAB3D*, as a recovery strategy. The general procedure of this algorithm is the following: the algorithm starts with the greedy phase until it finds a local minimum $c$. At this point, *GRG* stores the distance $dist(c, d)$, and switches to the random phase, as recovery strategy, where the node $c$ randomly selects one $u$ of its neighboring nodes, using the steps defined in *PAB3D*. If $dist(u, d) < dist(c, d)$, then the algorithm resumes the greedy forwarding, otherwise continues with *PAB3D*.

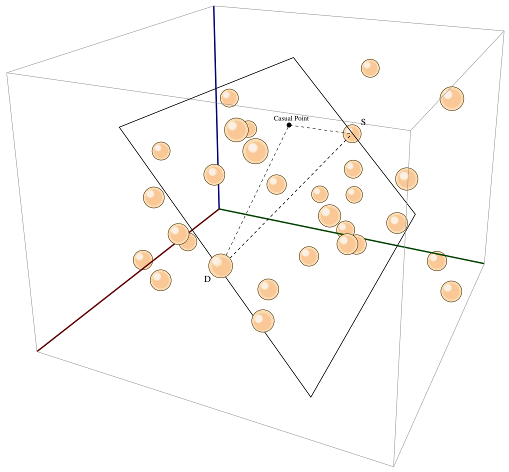


Fig. 4. Computing of a plane with *Projective face* algorithm

### G. Projective Face

The first extension of face-based strategy in 3D space provides using two orthogonal planes intersected to the line connecting source and destination. In [1] authors propose *Projective Face*, in which the nodes of the networks are firstly projected onto one plane that contains the line $sd$, with third point chosen randomly. Then, the *Face* algorithm is performed on this projected graph. If the routing fails, the nodes are then projected onto a second plane, that is orthogonal to the first one and also contains the line $sd$. Then, the *Face* algorithm is again performed. Fig. 4 shows an example of planes configuration in *Projective Face*. Note that, in this case (and in all the following algorithms), since the delivery rate is not guaranteed, the algorithm needs a local threshold value, $TTLF$ (*Time To Live Face*), in order to terminate the algorithm in case it not reached the destination. This is necessary because the algorithm can get stuck in a loop. More precisely, in this version of algorithm, $TTLF$ counter is started twice, once for the first plane and one for the orthogonal plane, obtaining a global threshold value, $TTL \leq 2 * TTLF$.

### H. CFace(3)

*CoordinateFace(3)* (*CFace(3)*), proposed in [10], uses another set of projection planes, that is composed by the planes $xy$, $xz$ and $yz$ for the projection of the nodes. *CFace(3)* works as follow. All nodes are projected on the first $xy$ plane ($node.z = 0$) and then the face routing is started on the projected graph. If the packet does not arrive at the destination ($TTLF$ has expired), the original coordinates of all nodes are

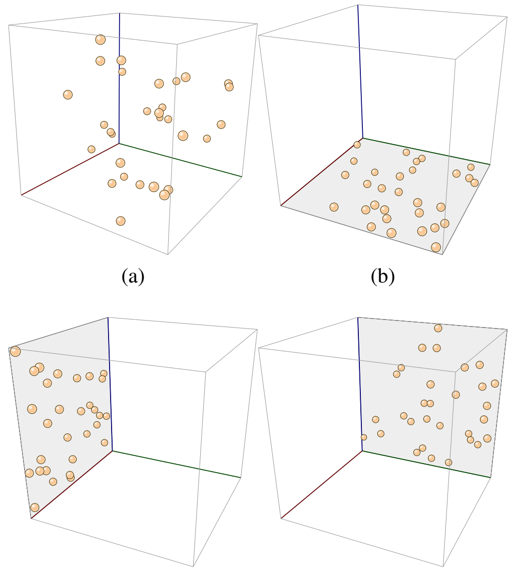


Fig. 5. Projection of graph nodes (a) on the three planes $xy$ (b), $xz$ (c) and $yz$ (d) in *CFace(3)* algorithm.

projected on the $xz$ plane ($node.y = 0$) and face routing is again performed. If again the packet does not arrive to the destination, the original coordinates of all nodes are projected on the $yz$ plane ($node.x = 0$) and face routing is again performed. If the packet does not arrive even with this last plane, the algorithm fails. Fig. 5 shows all the three projections on the coordinate planes. Note that, in this algorithm, $TTL$ is at most $3*TTLF$. As we will see, the delivery rate is typically greater than that of the *Projective face*, but with also a greater path dilation.

## I. Adaptive Least-Square Projective Face

Authors of [11] proposes three heuristics to modify and improve *Projective Face* algorithm. The new obtained algorithm is called *Adaptive Least-Squares Projective Face*, (*ALSP Face*). The three heuristics are:

- Least-Squares Projection (LSP) Plane;
- Adaptive Behavior Scale (ABS);
- Multi-Projection-Plane Strategy.

In the *Projective Face* algorithm, a third point is chosen randomly, together with the source and the destination points, to compute the first projection plane. Instead, *ALSP Face* chooses the third point adopting a mathematical optimization technique (first heuristic) for finding the best fitting plane to the set of neighbor nodes: using the current node, its neighbors up to two hops away, and the destination node, the initial projection plane is determined by using *least-squares error minimization* of the distance of the nodes to the plane, hence minimizing the sum

$$\sum_{i=1}^{m}(r_i)^2$$

where $r_i$ is the Euclidean distance from a point $i$ to its perpendicularly projected point in the least-squares projection plane (*LSP* plane), as seen in Fig. **??**. To maintain the local characteristic of the routing algorithm, authors propose that only the source, destination and the neighboring nodes within the 2-hops scope of the current node be selected as the set of points for computing the least projection plane [1] Then, nodes are projected on this plane and *Face* routing is performed. This *LSP* plane aims to have a less distorted projected graph so that the number of crossing edges can potentially be reduced. The second heuristic defines a parameter called *Adaptive Behavior Scale* (*ABS*) that is used to determine when recalculate the *LSP* plane, in order to ensure that the plan is always appropriate for the current node. The third heuristic uses a set of $N_s$ projection planes arranged in a fixed order about an axis. The algorithm switches between these planes, following the order, to disrupt any looping that may occur during routing. In [11] it is said that performing the face routing on the additional projection plane, significantly increases the delivery rate. Therefore, the third heuristic tries to increase the number of projection planes. But this is true only up to a certain point; even if it is true that the delivery rate is slightly increased, it is also true that the path followed by the packet becomes enormously long, because, for each projection plane, the threshold value (here, $TTLF$) is reset to its pre-set value. Such a high traffic for only one packet might not be acceptable in the real world. In this work we consider the third heuristic using only two additional planes, chosen as in *Projective Face*.

## J. Greedy-Face-Greedy

*Greedy-Face-Greedy* algorithm (*GFG*), also defined as *Greedy Perimeter Stateless Routing* algorithm (*GPSR*) for 2D networks, uses a combination of greedy method with face method. With *GFG*, a flag is stored in each data packet. This flag can be set into greedy-mode and face-mode (or perimeter-mode), indicating whether the packet is forwarded with *Greedy* or *Face* algorithm. The algorithm starts from $s$ with *Greedy*, setting the packet into greedy-mode and forwarding it. A node that receives a greedy-mode packet searches among its neighbors the node that is closest to the destination. If this neighbor exists, the current node forwards the packet to its, otherwise it marks the packet into perimeter-mode. *GFG* algorithm forwards the perimeter-mode packets performing the same planar graph as the *Face2* algorithm. Moreover, when a packet enters in perimeter-mode at node $x$, *GFG* set the line $xd$ as a reference line to the crossing with the faces and records in the packet the location of $x$ as the node when greedy forwarding mode failed. This information is used at next hops to determine whether and when the packet can be returned to greedy-mode: upon receiving a perimeter-mode packet, the current node $c$ first compares the location of $x$ as stored in the packet with the forwarding node's location. *GFG* returns in greedy-mode if the distance from the forwarding node to $d$ is less than the distance from $x$ to $d$. In this case, the packet is set into greedy-mode and the algorithm continues greedy progress towards the destination. Fig. 6 shows an example, where the

[1]for simplicity of implementation, in this work the 1-hop scope of current node was used for *LSP* plane computing

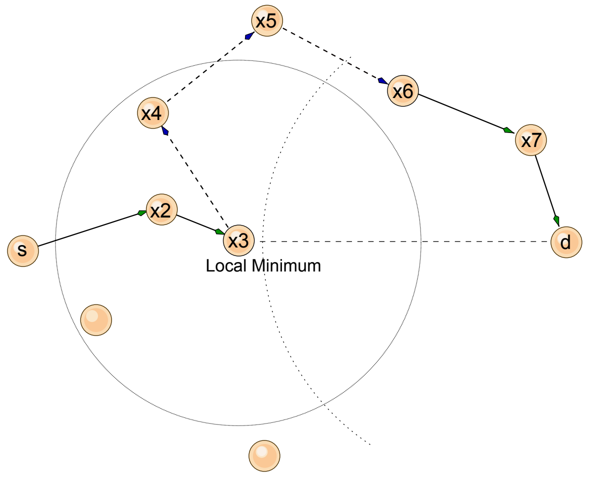


Fig. 6. Performing of *GFG* algorithm. Solid arrows represent greedy-mode forwarding, dashes arrows represent face-mode forwarding.

packet starts from $s$ and travels $x2$ and $x3$ in greedy-mode, stopping at $x3$, that is the local minimum. Then, from $x3$, the face-mode is started, and forwards the packet on progressively closer faces of the planar graph, each of which is crossed by the line $cd$. So, the packet reaches $x4$, $x5$ and $x6$ in face-mode. $x6$ is the closest node to $d$ than $x3$, so the packet can be returned to greedy-mode and reach $d$.

### K. PAB3D-CFace(1)-PAB3D

This algorithm, initially conceived in [10] as *AB3D-CFace(1)-AB3D*, starts with *PAB3D* algorithm. Once the local threshold $TTLR$ is passed and the algorithm reaches a local minimum, it switches to *CFace(1)*. *CFace(1)* traverses one projective plane, which is chosen randomly from the $xy$, $yz$, or $xz$ planes, starting from the node in which the algorithm is switched. At this point, $TTLF$ is initialized to 0 and *CFace(1)* restarts. If the destination is not reached during this phase and $TTLF$ is passed (a looping occurs), the algorithm goes back to *PAB3D* and $TTLR$ restarts at 0.

### L. PAB3D-CFace(3)

This algorithm starts as *PAB3D-CFace(1)-PAB3D*, but instead of going back to *PAB3D* if the phase in a projective plane fails, *PAB3D-CFace(3)* tries the other two projective planes, defined as in *CFace(3)*. This algorithm starts with the *PAB3D* stage and if the destination is not reached and the $TTLR$ is passed, the algorithm switches to *CFace(3)* using the first $xy$ plane. Again, if a loop occurred ($TTLF$ is reached), it switches to $yz$ plane, and finally the same process is done for $xz$ plane.

### M. LAR 3D

In the extension of *LAR* algorithm in 3D space, with three-dimensional coordinates of source and destination positions, the source node computes the expected zone for $d$, which is a sphere around $d$ of radius equal to $v_max * (t_1 - t_0)$ where $t_1$ is the current time, $t_0$ is the time stamp of the position information that c has about d, and $v_max$ is the maximum speed of the node in the network. This zone is used to define the 3D flooding area, which is the minimum size rectangular box with $ball(s, r)$ in one corner and the expected zone in the opposite corner. In our case, since we analyze static scenarios ($v_max = 0$), the expected zone is $ball(d, r)$.

### N. PAB3D-LAR

*PAB3D-LAR* was proposed in [12] and hybridize *PAB3D* with *LAR*. This algorithm tries to reduce the high path dilation of the *LAR* algorithm. All the combinations in *PAB3D-LAR* use the same partitions as in the *PAB3D* algorithm. The difference is that, while *PAB3D* selects only one of the $m$ candidates chosen from the neighborhood, *PAB3D-LAR* sends the packet to all those selected candidates which are within the rectangular box defined as in *LAR*.

## V. Performance Evaluation

In this section we provide a performance comparison of the above considered routing protocols. A 3D simulation environment is structured for the experiment and a common set of configurations is provided.

### A. Simulation Environment

Our simulation scenario consists of $n$ nodes placed in random positions in a 3D cubic space. The cube has 500 m of side length. Every scenario instance represents a connected networks, which means that each node can reach each other one. The node transmission range is set to 100 m. To simplify the simulation, we assume that the nodes are not moving, obtaining a static ad-hoc network. The motivation of this choice is twofold: (a) the need to consider initial essential parameters for the comparison (i.e. the minimum path) that would change if the nodes move, and (b) the assumption that, with the considered routing procotols, the route of each packet is traveled in a very little time, during which the nodes of the networks are essentially still. The first comparison aims to get a comprehensive look at the behavior of the forwarding algorithms under dinamyc number of nodes, ranging in {50, 100, 150, 200}. Second comparison analyzes the effect of varying the threshold values *TTLR* and *TTLF* of respectively randomized-based and face-based forwarding algorithms. The third comparison applies all the forwarding algorithms onto classes of graphs in which the length of the shortest path from source to destination ranges in 1-3 hops, 4-6 hops, 7-9 hops and 10+ hops (the notation *a-b* indicates that the shortest path length in a-b class graph can assume values from $a$ to $b$ including).

### B. Results

We discuss here the outcome of the experimentation and analyze the protocols' performances on different settings.

*1) Comparison with dynamic number of nodes:* First comparison performs considered routing protocols in a set of network istances consisting of 50, 100, 150, 200 nodes. The results are shown in Fig. 11 and 8. From the results in Fig. 11, we see immediately that deterministic progress-based forwarding algorithms have the lowest delivery rate, less than 60% in the case of 150 nodes. This means that in this scenario (150 nodes) there is more chance of finding local minima. Instead, in the case of 200 nodes, we can see an increment of the delivery rate. This suggests a reduction of local minima with a growth of nodes' number starting from 150, in a cube of 500 meters of side length. The *PAB3D* and *G-PAB3D-G* algorithms obtain better results in delivery rate, with a peak in the case of 50 nodes (90%), as seen in Fig. 11a. Face-based routing algorithms (*Projective Face*, *CFace(3)*, and *ALSP Face*) have generally better results than the previous ones, in terms of delivery rate. This overcoming depends on the fact that these algorithts have more possibility to find the destination node, since *TTLF* resetting. *CFace(3)* can be defined as *Projective Face*, only choosing two plans that are *xy* plane and *xz* plane, and with the addition of the *xz* plane. So, since the algorithm has three chances to find the destination instead of two as in *Projective Face*, the delivery rate is a little higher. The hybrid algorithm, *GFG*, reaches better results compared to *ALSP Face*, regarding delivery rate. This is caused probably by the greedy method phase that, at first, carries the packet towards the destination and then, if a local minimum is found, the face method phase starts, but with less search space. *LAR* and *PAB3D-LAR* are partial flooding-based algorithms. *LAR* has relevant traffic and does not show a very high delivery rate. Not even *PAB3D-LAR* presents a good delivery value, but has almost half of path dilation than *LAR*. From what we can see in Figs. 11c, 11d, the transition from 150 nodes networks to 200 nodes ones increases the delivery rate, due to the increment of the node density and the restricion area includes many more nodes.

*2) Comparison with dynamic threshold values:* This performance test has the purpose of assessing the effect of the threshold values respectively for randomized-based and face-based algorithms, that is to see how effective is to increase the *TTLR* and *TTLF* values to increase delivery and path performance. In this test *TTLR* and *TTLF* are dynamic, while the number of nodes *n* is not. Therefore, to get a fair treatment of the various instances and to allow each algorithm reach all their thresholds values (i.e., *CFace(3)* fails after three times *TTLF*), the *TTL* value is calculated equal to *2 * TTLR* for randomized case and to *3 * TTLF* for face case.

*3) Comparison with dynamic min path length:* This section shows the results relating to the application of all the routing algorithms considered in classes of graphs in which the length of the shortest path source-destination is respectively 1-3, 4-6, 7-9, and >10 hops. These results can be seen in Fig. **??**. Note that for the first class (1-3 hops, in Subfig. **??**), all the algorithms have a high delivery rate and a small path dilation. This happens because there is very little chance that in a so few hops distance there are holes (local minima). When it starts to move to longer paths, deterministic progress-based strategies are the first to degenerate, because, as the destination is more distant, there is a greater probability to find holes. As seen in Subfig. **??**, progress-based algorithms reach about 10% of delivery rate. *PAB3D* and *GRG* perform well up to 4-6 length of minimum path, but from this point the delivery rate decreases. This is because the increasing number of links to choose from source to destination reduces the probability of choosing the righ path. However, the path length of the delivered packet remain short (see Subfig. **??**, PAB3D and GRG histogram). The performance of face-based and hybridized algorithms are also good for high lengths, due to the fact that they succeed in reaching the destination within the $TTLF$ or $TTL$, despite the increasing of the distance from source to destination. However, the length of path performed is too high, due to an increment in the number of crossing links during the travel. [1]

## VI. Conclusion

This survey had depeened, in many ways, the problem of position-based routing applied on three-dimensional networks. Firstly, the reasons of using an approach based on the position is discussed, and then the problems and limitations of the algorithms that use these techniques.

Deterministic memoryless progress-based strategies can perform well in very dense networks, but not in a sparse networks, due to the problem of local minima. Algorithms that use this strategies, for instance *Greedy*, may be used in combination with other algorithms, as seen, in order to reduce effectively the number of nodes traveled.

The random component in a forwarding decision offers a better chance to reach the destination. In this work, we have recovered all the randomized-based algorithms and unified them into a single algorithm called *PAB3D*, with different input parameters. With a better combination of possible parameters, *PAB3D* reaches a delivery rate of %80, with a path length of at most three times the minimum path length in the considered scenarios. Hybridizing it with *Greedy*, getting *GRG*, there is not a significantly reduction of path dilation.

## Appendix A
## Appendix 1

Appendix one text goes here.

## References

[1] J. O. G. Kao, T. Fevens, “Position-based routing on 3d geometric graphs in mobile ad hoc networks,” pp. 88–91, 2005.

[2] G. G. Finn, “Routing and addressing problems in large metropolitan-scale internetworks,” 1987.

[3] X. L. I. Stojmenovic, “Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks,” vol. 12 (10), pp. 1023–1032, 2001.

[4] H. S. E. Kranakis and J. Urrutia, “Compass routing on geometric networks,” pp. 51–54, 1999.

[5] L. K. H. Takagi, “Optimal transmission ranges for randomly distributed packet radio terminals,” vol. 32 (3), pp. 246–257, 1984.

[6] K. S. K. Yamazaki, “The proposal of geographical routing protocols for location-aware services,” vol. 87(4), 2004.

[7] L. K. R. Nelson, “The spatial capacity of a slotted aloha multihop packet radio network with capture,” vol. COM-32, pp. 684–694, 1984.

[8] P. M. P. Bose, “Online routing in triangulations,” pp. 113–122, 1999.

[9] B. N. B. T. Fevens, A. E. Abdallah, “Randomized ab-face-ab routing algorithms in mobile ad hoc networks,” 2005.

[10] A. Abdallah, T. Fevens, and J. Opatrny, “Randomized 3d position-based routing algorithms for ad-hoc networks,” pp. 1–8, 2006.

[11] J. O. G. Kao, T. Fevens, “3d localized position-based routing with nearly certain delivery in mobile ad-hoc networks,” 2007.

[12] J. O. A.E Abdallah, T. Fevens, “High delivery rate position-based routing algorithms for 3d ad-hoc networks,” vol. 31 (4), pp. 807–817, 2007.

## References

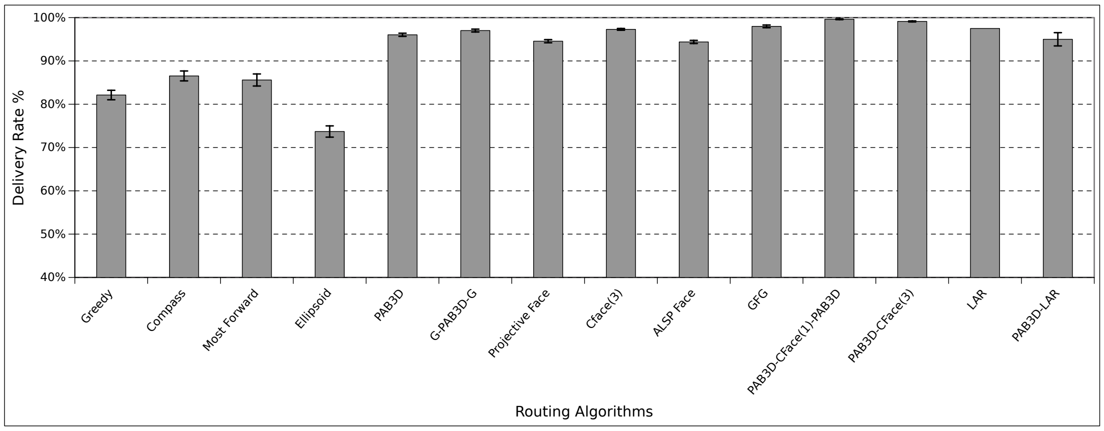
[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

PLACE PHOTO HERE

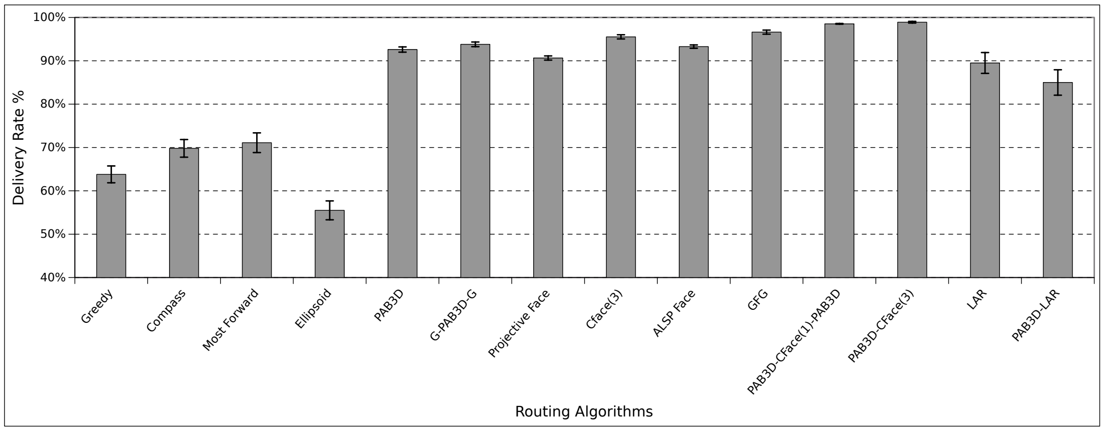
**Claudio Enrico Palazzi** Biography text here.

**Armir Bujari** Biography text here.

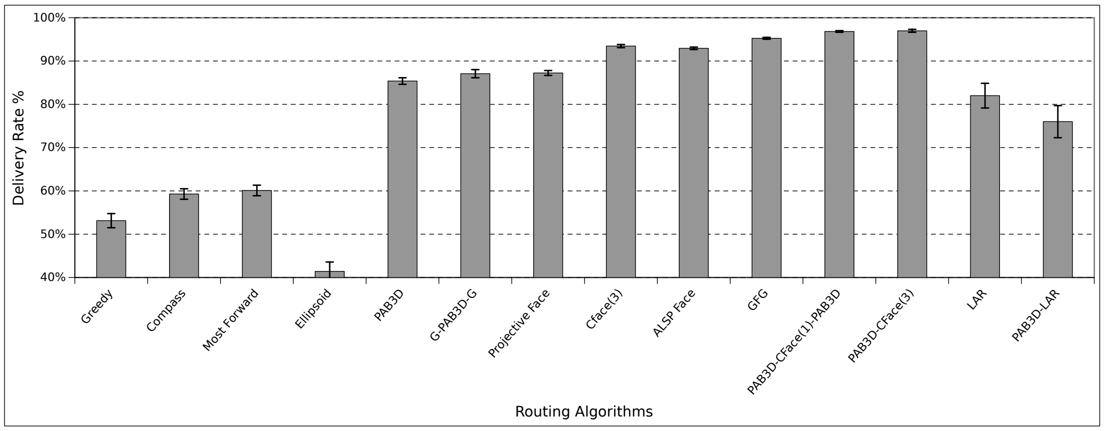
**Daniele Ronzani** Biography text here.



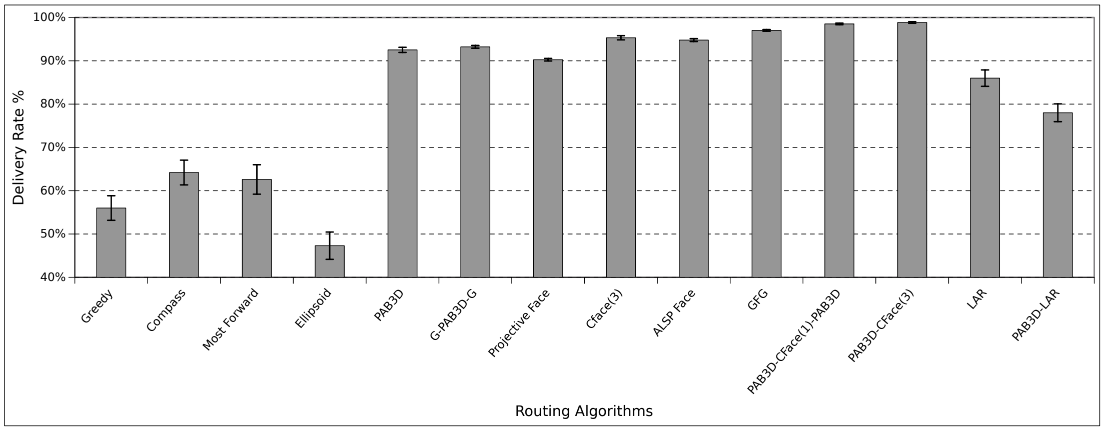
(a)



(b)

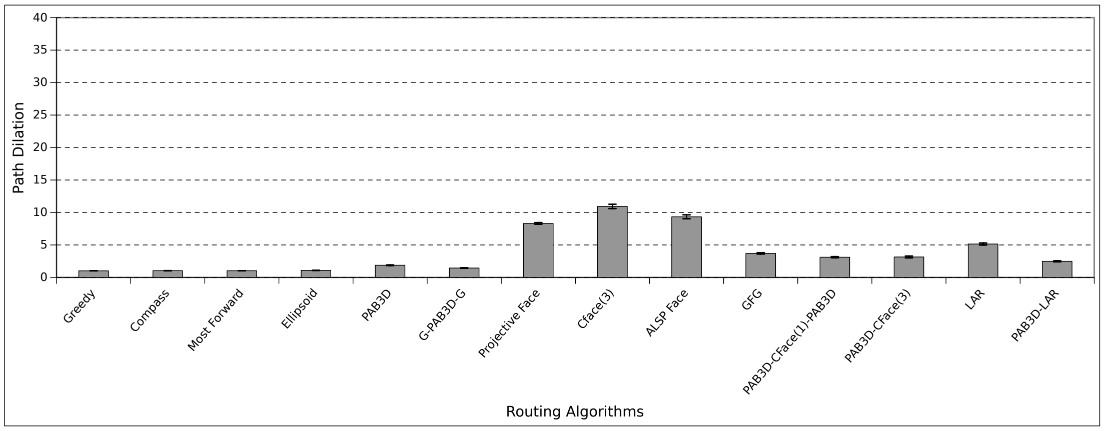


(c)

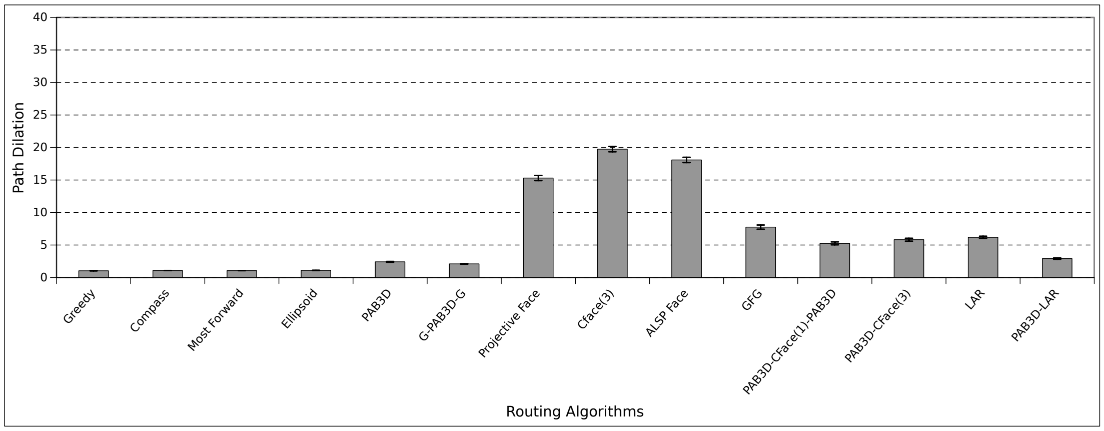


(d)

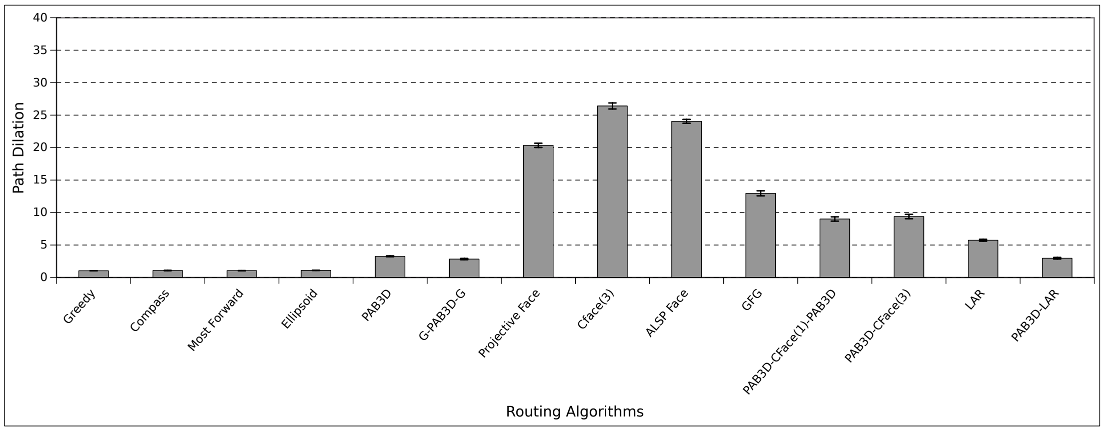
Fig. 7. Results of delivery rate



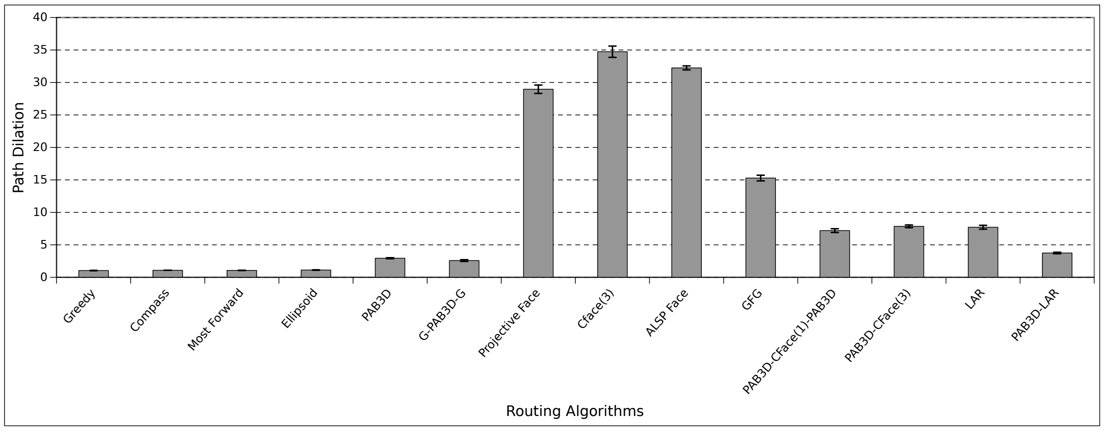
(a)



(b)

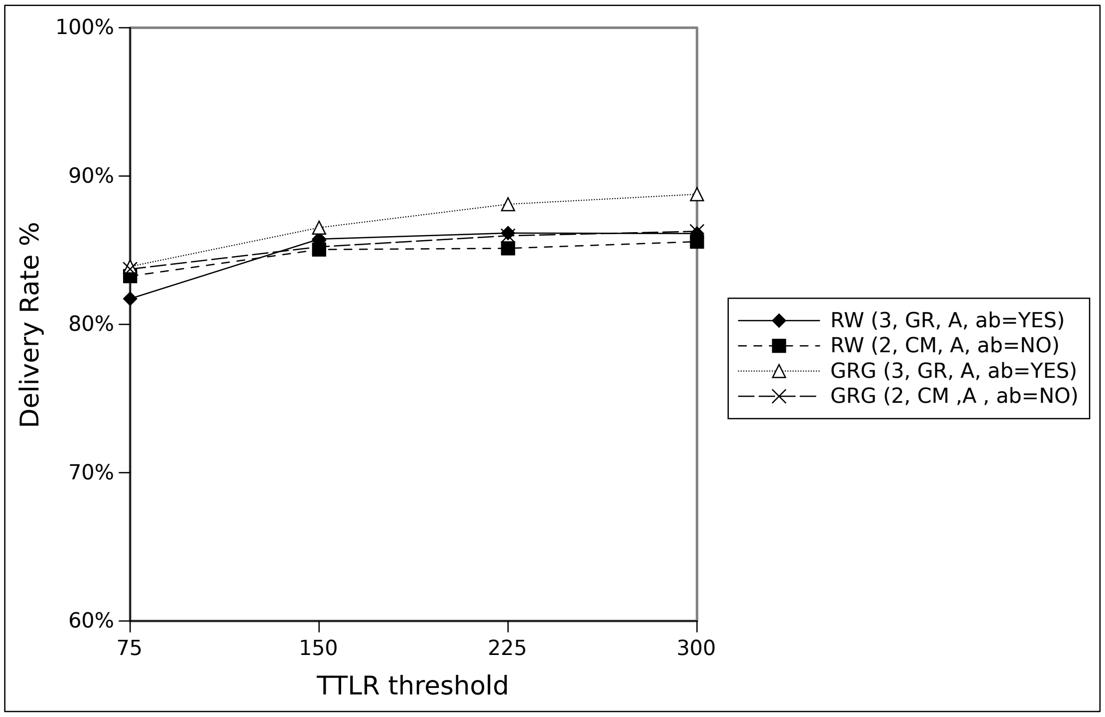


(c)



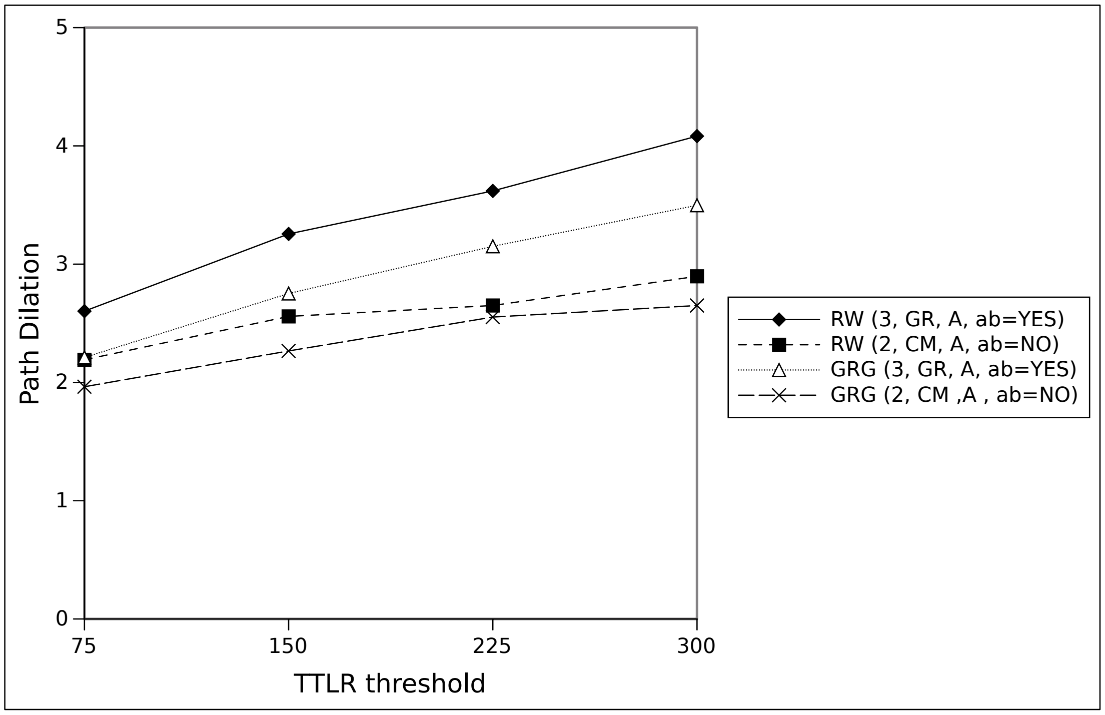
(d)

Fig. 8. Results of path dilation



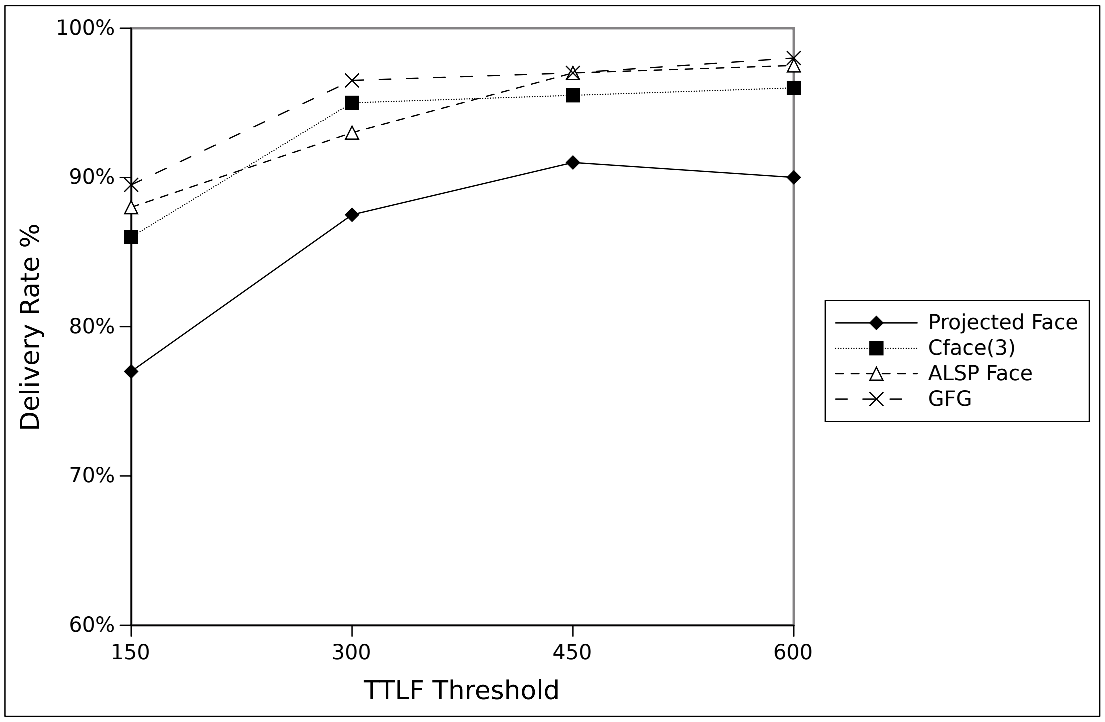
Delivery Rate %
100%
90%
80%
70%
60%
75
150
225
300
TTLR threshold
RW (3, GR, A, ab=YES)
RW (2, CM, A, ab=NO)
GRG (3, GR, A, ab=YES)
GRG (2, CM ,A , ab=NO)

(a)

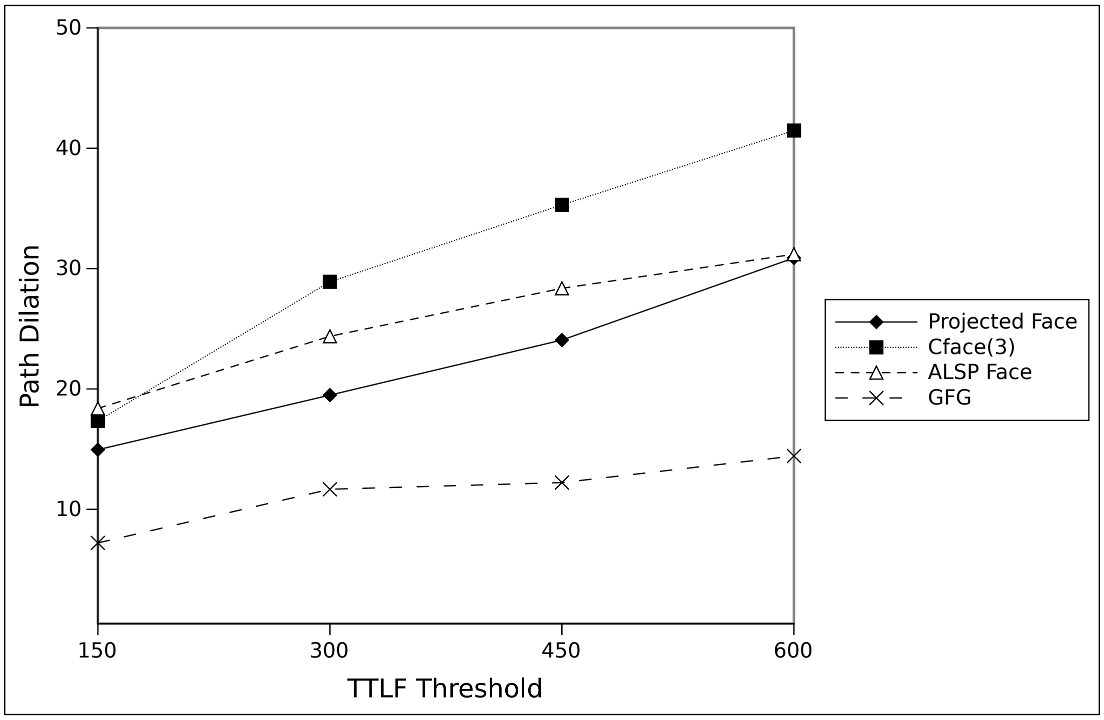


Path Dilation
5
4
3
2
1
0
75
150
225
300
TTLR threshold
RW (3, GR, A, ab=YES)
RW (2, CM, A, ab=NO)
GRG (3, GR, A, ab=YES)
GRG (2, CM ,A , ab=NO)

(b)

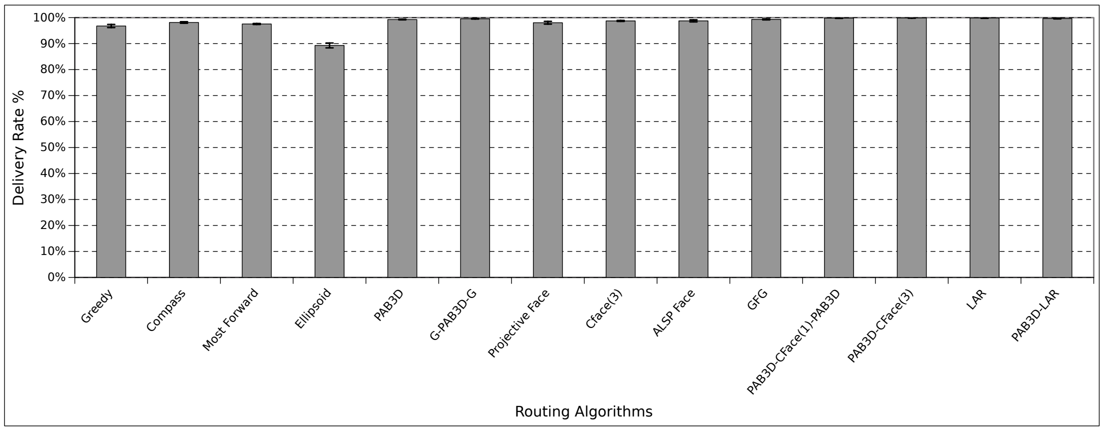


(a)

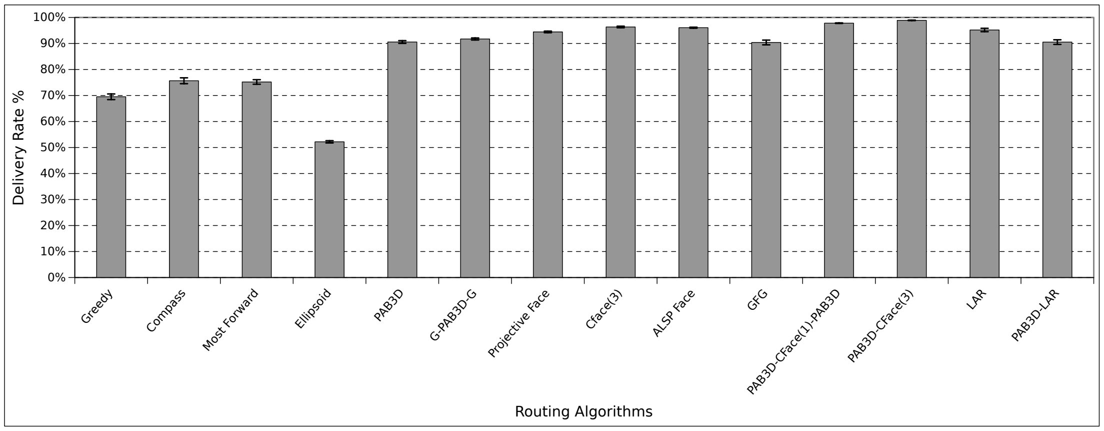


(b)

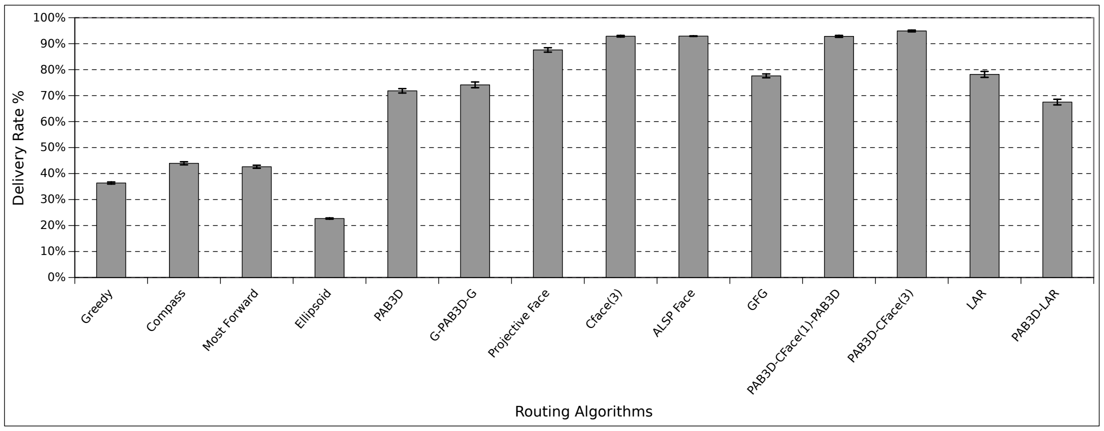
Fig. 9. Results of path dilation



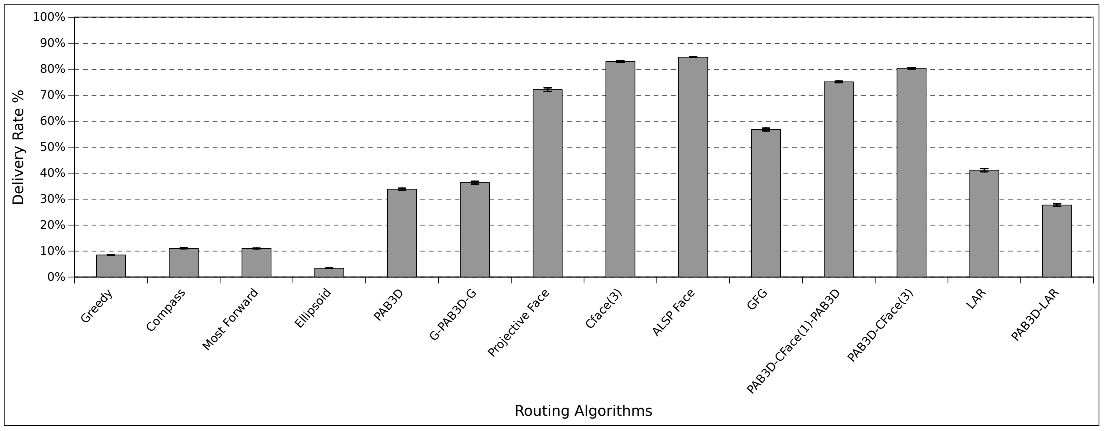
(a)



(b)

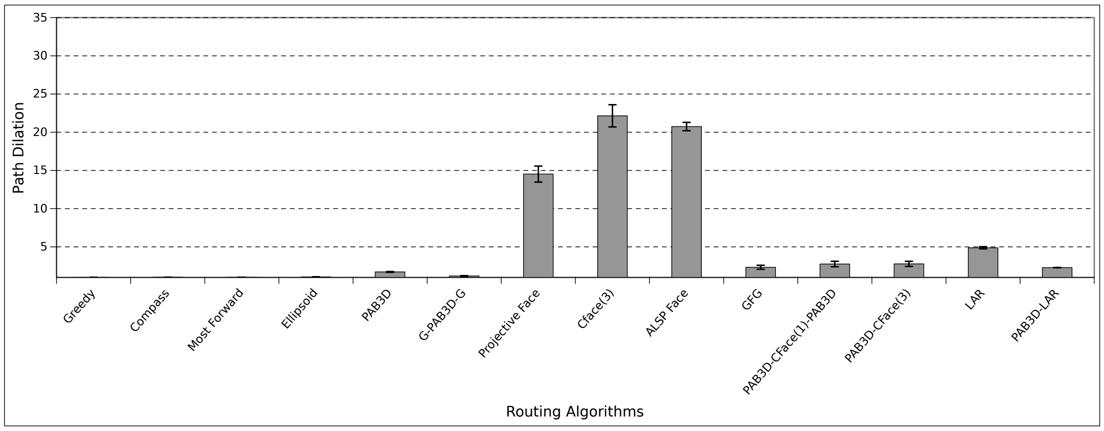


(c)

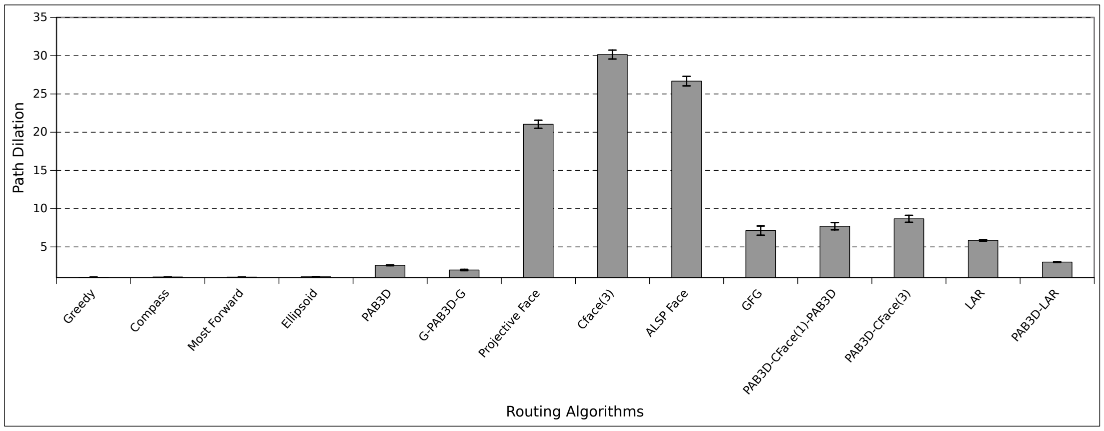


(d)

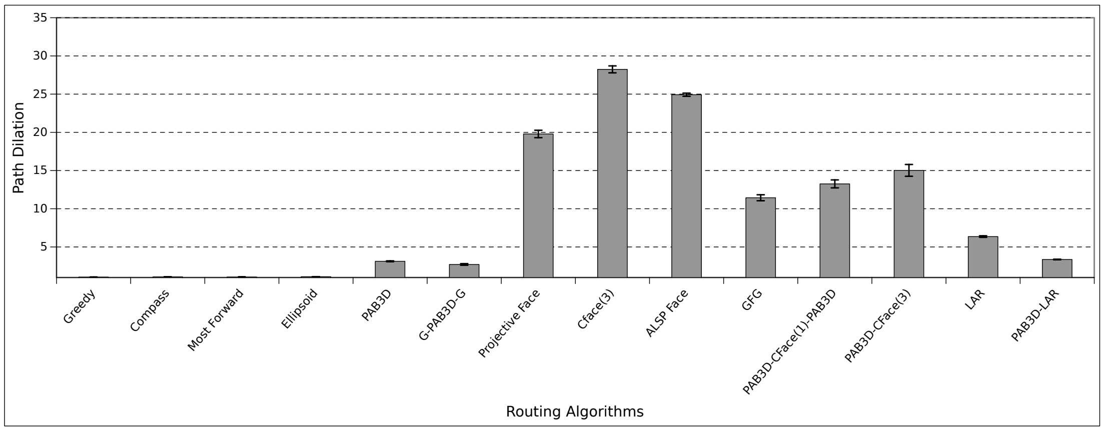
Fig. 10. Results of delivery rate



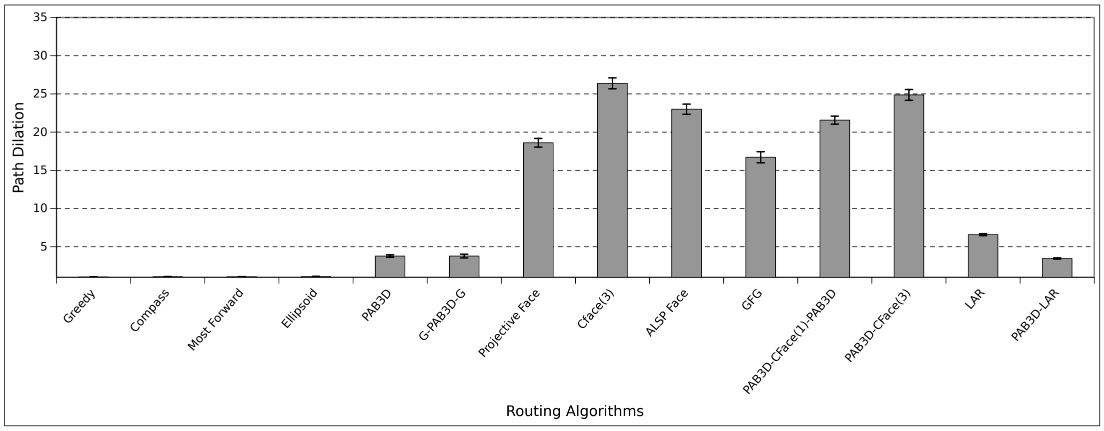
(a)



(b)



(c)



(d)

Fig. 11. Results of delivery rate