Generating Arrival Routes with Radius-to-Fix Functionalities

Valentin Polishchuk
ITN, Linköping University, Sweden

***Abstract*—We investigate Radius-to-Fix (RF) functionality – a future operational concept present in several specifications of Performance-Based Navigation (PBN), including those used for terminal area operations. We show how to generate operationally feasible arrival routes using RF segments, balancing two conflicting objectives: (horizontal) efficiency of the flight paths in the produced STAR tree topology and keeping the overall traffic picture amenable to monitoring. The balance is achieved and controlled by essentially one user-tunable parameter in our model – the minimum required separation distance between route merge points. This makes our approach simple and easy to implement, which we demonstrate by running our algorithm for runways in Stockholm Arlanda airport. At the same time, our general scheme is directly extendable in several ways, allowing one to attach add-ons relevant in various specific application areas.**

Keywords: PBN, Radius-to-Fix functionality, STAR, TMA

I. Introduction

ICAO Performance-based Navigation (PBN) program is one of the key enablers of the upcoming and more distant future air transport modernization. PBN consists of Area Navigation (RNAV) which enables flying along an arbitrary path in the area covered by ground- or space-based navaids, and Required Navigation Performance (RNP) which adds to RNAV the capability of onboard monitoring of the performance achievements and alerting the crew about requirements that are not fulfilled during operation [1,2]. PBN gets highest priority for international harmonization by ICAO's Global Air Navigation Plan (GANP) complemented by the Aviation System Block Upgrades (ASBU); in particular, ICAO's resolutions from the 36th and 37th assemblies urge all states to implement PBN in accordance with the PBN Manual [3].

PBN in TMAs. Airspace users are informed about PBN requirements via a family of navigation specifications, such as RNAV-x or RNP-x where x is a number signifying the performance level. The specifications differ, inter alia, by the set of phases of flight which they cover. For instance, RNP-1 is limited to use on STARs/SIDs, initial/intermediate segments of IAPs and missed approach after the initial climb phase. On the other side of the spectrum of applicability breadth, there stands *Advanced RNP (A-RNP*) specification, which emerged after wide-ranging debates (in particular, in Europe, A-RNP is replacing both B-RNAV and P-RNAV), and encompasses all phases of flight including the Final Approach. Overall, for a transition airspace, RNAV-1, RNP-1 and Advanced RNP are the most essential available specifications [3].

A. Radius-to-Fix

PBN specifications differ also by embedded functionalities. RNP specifications featuring extra functionalities provide higher design flexibility (in comparison with those whose functionalities are scarce) which is especially useful in complex, high-density airspaces, such as TMAs. The *Radius-to-Fix (RF)* functionality is available in RNP-1 and higher ICAO navigation specifications, i.e., RNP APCH AMC 20-27 and AMC 20-28, RNP AR APCH AMC 20-26 and RNP 0.3 (used primarily for helicopter operations); in particular, A-RNP is *required* to have RF [5], while for other specifications RF functionality is optional. Concrete conditions of RF use are detailed in [3, RF Appendix to Volume II of Part C].

Track predictability and repeatability are some of the main requirements that led to the introduction of A-RNP; RF functionality provides these in the turning segments in TMA, and this is why RF had been made mandatory for A-RNP. The requirements are even more crucial in a TMA environment with its limited capacity and higher need for the predictability, motivated by the strict lateral separation standards and changing altitudes of the flights. The enroute counterpart of RF is Fixed-radius Turn (FRT).

B. Paper outline

In this paper we develop a precise mathematical framework for designing arrival traffic merge routes and provide the first algorithmic solution to the problem of creating STARs using RF segments. In Section II we recap known mathematical properties of *curvature-constrained paths* which consist of sequences of straight-line segments (standard point-to-point legs, in ATM parlance) and circular arcs (the RF legs). Section III defines the input and output for of our problem, and Section IV builds up the constraints for the model of operationally feasible arrival tree (STAR tree) for merging traffic from entry points to the runway. In Section V we give our algorithmic solution approach for constructing the STAR tree. Section VI presents results of implementing our algorithm for Stockholm TMA (Arlanda airport). Section VII concludes the paper.

C. Related work

Planning motion with bounded radius of turn is relevant in many domains (also outside ATM) and has a long history of research (the earliest reference we are aware of is [6]). The first characterization of shortest bounded-curvature paths was given by Dubins [7] (we restate it below in Theorem 1); the result has been subsequently reproved and extended by control theory techniques in [8,9]. Constructing trees with curved edges has also been researched in visualization works where logarithmic spirals were used to produce nicely looking trees illustrating, e.g., shipment of goods [10,11]; the goods were routed to a *point* and curvature of paths within the tree was not a constraint.

This research is part of the ODESTA (Optimal Design of Terminal Airspace) project funded by the Sweden's innovation agency VINNOVA and LFV.

In general, a large body of ATM research is devoted to terminal area operations. However, to the best of our knowledge, only two papers looked at the specifics of *merging* paths in a STAR tree topology and no prior work considered using *curved legs* in path generation. Below we survey the earlier papers that are most closely related to our study.

Krozel et al. [12] gave an algorithm for finding paths that avoid "frequent" and "sharp" turns. The algorithm produces a *polygonal* path, i.e., path composed of straightline segments turning at vertices. One of the applications for the paths, mentioned in the paper, is arrival paths for TMA; still, the issue of merging the paths was not considered.

In [13,14] Michalek and Balakrishnan evaluated stochastic properties of (pre-determined or modified) TMA routes; the routes were not assumed to form a merge tree. In [15], an algorithm was given to produce arrival routes through the TMA; the routes, however, were mutually disjoint and all ended at metering fixes on a ring around the airport -- the merging of the routes from the fixes to the runway(s) was assumed to be done inside the ring and was not considered in the paper (the modeling in [13,14] was the same in this respect). The only works we are aware of that did consider merging the routes into the arrival tree are [16,17]; still, the routes were polygonal and merged to a point, without specifying the direction before landing.

II. Preliminaries on Curvature-Constrained Paths

We begin with some terminology and results related to bounded-curvature motion.

Notation. Let r denote the smallest allowable turn radius in a flyable path. All circles and circular arcs in this paper will have radius r; we will thus often omit the specification of the radius and simply say *circle* and *circular arc* (or even simply *arc*) to mean "circle of radius r" and "arc of radius-r circle" respectively. We will use the shorthand *direction of the path at a point* to mean "direction of the tangent to the path at the point" (we consider only smooth paths, having tangents at all their points). A *configuration* is a pair (p,P) where p is a point and P is a direction. Two mutually tangent circles whose common tangent passes through p in the direction P will be called the *tangent circles* or simply the *circles* of (p,P) and denoted by $R(p,P)$ and $L(p,P)$; the clockwise (resp. counterclockwise) traversal of R (resp. L) goes in direction P at p (refer to Fig. 1). Finally, unless otherwise stated, we will say simply *segment* for "straight-line segment".

Curvature constraint. Central to our investigations is the requirement that a flyable path should never have radius of turn smaller than r; such paths are known as *bounded-curvature* or *curvature-constrained paths*, and we will use *cc* as a shorthand for "curvature-constrained". Formally, the turn constraint means that if s,f are two points on a cc path and the distance between s and f along the path is d, then the angle between the directions of the path at s and f is at most d/r (Fig. 1); an equivalent definition is that at any point p of a cc path, the path locally does not enter the circles of (p,P) where P is the direction of the path at p. Cc paths are planned not just between points, but between configurations -- since the curvature bound essentially gives a constraint on the second derivative of the path, the boundary conditions for a cc path must specify not only the path endpoints, but also the first derivatives, i.e., the directions of the path at its endpoints.

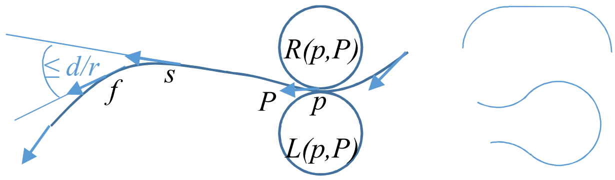

Figure 1. A cc path has no sharp turns. On the right, the two forms of a shortest cc path are shown (in the case in the bottom, the middle arc necessarily spans more than 180°).

Dubins paths. The cornerstone result in curvature-constrained motion planning, first obtained by Dubins, states that optimal cc paths use only segments and arcs:

Theorem 1 [7]. A shortest cc path either consists of an arc followed by a segment followed by an arc, or from a sequence of three arcs (Fig. 1, right).

Moreover, it was shown that in the second form of cc path the middle arc is at least half the circle:

Theorem 2 [18]. In a shortest cc path consisting of 3 arcs, the turn along the middle arc is larger than 180°.

As a little twist to the theory of Dubins paths, useful for our further development, we note that it is also possible to consider a cc path between a point (without specifying the initial direction) and a configuration. In particular, the following is implied by results in [18] (where it was investigated how cc paths look when at one of the endpoints the path is free to have any direction) and [19] (where the *shortest cc path map*---the description of how the paths to a fixed final configuration vary depending on the initial configuration---was built):

Theorem 3 [18,19]. Let (f,F) be a configuration, and let s be a point outside the circles of (f,F). Then the shortest cc s-f path consists of the tangent from s to one of the circles of (f,F) followed by the arc of the circle.

III. Problem Formulation

Our ultimate goal is to produce STARs that connect TMA entry points with the runway via cc paths. This section gives exact definitions of the input and output of our problem; the next section elaborates on the constraints.

Graph theory preliminaries. We recap some graph-theoretic notions and state terminological assumptions which we make for ease of exposition. A *tree* is a graph without cycles; a *rooted tree* has one of its nodes designated as the *root*. We look for a tree embedded in 2D; i.e., any node of our tree is a point in the plane, and edges of the tree are curves in the plane connecting pairs of nodes (in fact, each edge of our graph will be either a segment or an arc). We will deal only with rooted trees, and will say just *tree* to mean "rooted tree". In any tree, there is a unique path between any two nodes, and unless otherwise stated by a *path* from a node in the tree we will understand the path from the node to the root. The *parent* of a node v is the next node on the path from v (e.g., the root is the only orphan node), and a *child* of v is a node for which v is the parent. We will assume that the edges of the tree are directed along the paths to the root (this is contrary to the standard, but for us it is more natural to direct the paths towards the root because our trees will be rooted at the runway). The *in-degree* of a node is the number of its children; we sometimes say just *degree* to mean "in-degree". A *leaf* is a node of degree 0, i.e., a childless node. A tree node that is neither a leaf nor the root is an internal node. Often we will call an internal node whose (in-)degree is larger than 2, a *merge node* or simply a *merge* (because paths to the root merge at such node).

Definition. A curvature-constrained tree is a tree in which every path is a cc path.

As with the cc paths, we will use *cc tree* as a shorthand for "curvature-constrained tree", and often will even omit the modifier "cc" since all our trees will be cc trees.

Input and output. We are now ready to specify what is given and sought in our problem.

Given:

1. The set of n entry points to the TMA. Each entry point is identified with its geographical location in 2D (lon,lat). Also given is the traffic load of each point (the number of aircraft that arrive via it). The other info about the point (its altitude, type, etc.) is not part of our model.
2. The runway to which the traffic from the entry points must be routed. Specifically, we are given the coordinates of the RWY threshold (lon,lat) and the RWY direction (denoted by F). All other characteristics of the RWY (length, surface, etc.) are irrelevant in our model.

Find:

Cc tree whose leaves are the entry points and whose root is the runway.

IV. FURTHER MODELING

In this section we work out several properties of cc trees. Some properties are proved as inherent to any optimal cc tree, while others are developed, step-by-step, taking into account specifics of the ATM domain; altogether, they are viewed as the constraints that our desired solution will satisfy.

The separation distance. Before delving into the constraints description, we introduce the user-tunable *separation parameter* L which will show up in several of our constraints -- defining minimum distance in the RWY direction, enforcing separation between RF legs, etc. While in principle a separate distance parameter could be used in each of the constraints, for the simplicity of our model we use the same parameter L for all of them. (Our solution does not depend on this restriction, so it can be lifted straightforwardly.)

Segments and arcs only. We show that similarly to cc paths, optimal cc trees are comprised from segments and arcs. The following result follows directly from Theorem 1, but we state it here separately, for ease of reference:

Theorem 4. For any cc tree T there exists a cc tree T^*, with the same root and leaves, whose edges consist of segments and arcs, and such that any path in T^* is not longer than the corresponding path in T.

Proof. Consider any part of the tree T between two consecutive merge nodes a and b (i.e., the subpath a-b in the tree). By Theorem 1, the subpath between a and b may be replaced by a path consisting of segments and arcs without lengthening the path. Similarly, the whole tree T may be converted to T^* as in the theorem statement. q.e.d.

The EU PBN Implementation Handbook [4, p.53] defines a flight path as "a varied sequence of connected straight and curved path segments." Although in principle the curved segments might potentially be represented by arbitrary curves, Theorem 4 allows us to restrict attention only to segments and arcs (RF legs) as components of the STAR merge tree.

No segment-segment. Clearly, two segments may not be adjacent along a cc path because the curvature constraint would be violated at the segments common endpoint; in fact, the main novelty of this paper is that we step away from (the golden standard in route planning of) employing *polygonal* paths (sequences of segments with turns from segment to segment happening at the vertices -- infinite-curvature non-differentiable points) and use smooth paths instead.

A trivial "solution". Without any further constraints, the optimal solution to our problem is given by the tree with n edges that directly connect the entries to the runway threshold (Fig. 2) -- in such a tree, every leaf-to-root path is as short as is possible and hence all planes would fly the best possible routes (if they only could fly so). Naturally, such a solution is not what one is looking for, at the very least for the following two reasons:

1. Sharp turns. The planes may need to take a prohibitively sharp turn from a tree edge onto the runway (if an edge is not aligned with the RWY direction).
2. Control impossibility. Even if the planes could actually make the sharp turns at the runway threshold, such STARs would be very hard to control, as all n paths would merge at a single point -- a discouraging situation for traffic control (especially since in this case the merge would happen immediately at the runway).

Degree-2, separated merges [16]. To deal with the high degree merge, it was suggested in [16] that every merge node must have (in-)degree 2. This constraint alone would not fully resolve the issue though, since the constraint can be satisfied by replacing the high-degree node with a spanning tree of n nodes placed very closely to each other and connecting the entries to the new nodes. To deal with such a "cheat", [16] introduced the separation minimum between merges: the distance between merge nodes must be at least L.

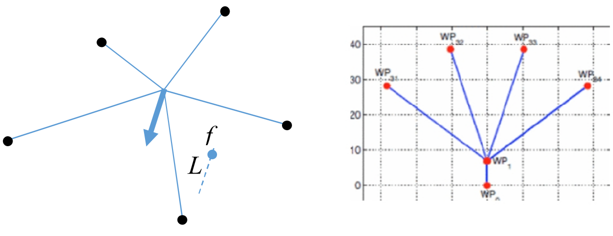

Figure 2. Entrys (black) would connect to the RWY (thick arrow) with sharp turns and a high-degree merge; to align aircraft with the RWY we require the paths to pass through (f,F). Right: the analogous baisc tree topology from Fig. 5 in [17].

RWY alignment. The above degree and separation bounds were used in [16] to find trees that merge traffic to a *point*, not to a *configuration* as relevant for us (this is expected, since [16], just as all prior work, considered polygonal trees, not cc trees). We set the RWY direction, F, as the final direction for the paths in our cc tree. Specifically, we require that any path gets aligned with the runway direction at distance L prior to the touchdown (on the final approach): we define the point f=shift(threshold, -F,L) to be the threshold shifted by L in direction opposite to F (see Fig. 2). We require that all paths in the tree end at the configuration (f,F). In what follows we will treat f as a merge node of the tree (even if no paths actually merge at it) because this is an important point and requires perhaps no less attention from the control than a "real" merge.

The next solution: UPR. While we do specify the final direction F for the paths (i.e., the direction of the tree edge incident to the root), we do not specify any direction for the tree edges at the leaves; we envision that the aircraft are able to align their flight direction with the edge of the STAR tree incident to the entry point by appropriate maneuvering in advance of approaching the TMA. It is reasonable to assume that the entries are situated sufficiently far from the runways (for a concrete bound, say, further than nL), to allow the air traffic controllers (ATCOs) get hold of the aircraft on the STAR (otherwise the whole transition airspace purpose would be defeated); in particular, in our terms, the entries are outside the circles of (f,F). By Theorem 3, shortest cc paths from the entries would all merge to the circles of (f,F), altogether forming a tree which would be ideal for the airlines, as each path will be a user-preferred route (UPR) – it will be as short as possible given the curvature constraint (Fig. 3).

One merge per arc, at an endpoint. Even though RNP tracks in general require less attention (vectoring) from controllers, the UPR tree would not resolve the issue of traffic control, as ATCOs would undoubtedly protest having all n paths merge on 1 (or at most 2) arcs. Moreover, if the merges happen in the middle of an arc, different aircraft will follow different sub-arcs of the same arc (this, in fact, will remain an issue even if only two paths merge interior to an arc), which, again, may be confusing for the controllers (especially since the different merge points would be too close on the arc, simply because the arc itself is not long). We therefore require that any arc takes part in at most 1 merge and that the merge is an endpoint of the arc.

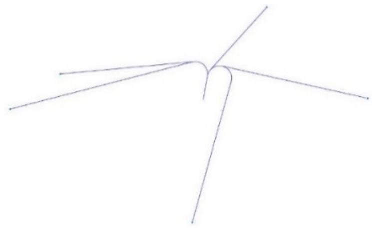

Figure 3. A UPR cc tree.

One path straight at a merge. Since we require that any merge happens only at an endpoint of an arc, the outgoing edge of any merge node must be straight. We go a bit further and request that one of the incoming edges to any merge point is straight as well. In other words, we require that at any merge point, one of the merging paths goes straight through the point and the other merges to it via an arc. This constraint may likely be lifted in the future when the high precision of RNP becomes routine (in fact, some to-date STARs already feature not only two arcs merging at a point, but also higher-degree merges); nevertheless, respecting the constraint may help ATCOs to oversee the traffic because for the straight path there is "less to worry about", and the attention may be focused on merging in the arc. (It is interesting to note that the requirement actually holds true in some existing STARs, e.g, in Stockholm Bromma airport; see Fig. 4).

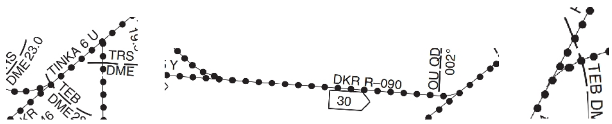

Figure 4. Zoom-in on Bromma STAR merges (from LFV AIP [20]).

A. Summary

Altogether, our rules for the design of the RF-enabled STAR merge trees are as follows: end at (f,F); segments and arcs only; no segment-segment; at least L between merges; 2 paths at each merge -- one going straight, the other merging with an arc (so there's 1 merge/arc, at the arc endpoint).

V. Algorithmic Solution Approach

In the previous sections we developed the input, output and constraints for our problem; however, we have not specified the objective function. It is hard to define a single objective function that would reflect the conflicting preferences of all the stakeholders: the airlines would like to fly along the UPR tree, while the controllers might prefer a a "Steiner tree" spanning the entries and the runway – a tree that "does not occupy much space" We therefore do not attempt to grasp the elusive "correct" objective function, and instead greedily build a tree which has short entry-to-runway routes, while satisfying the constraints outlined in the previous section.

Specifically, we assume that the entries are sorted 1...*n* by the decreasing load, and connect the first (i.e. highest-load) entry to the runway using the shortest cc path. We then connect the second entry with the shortest possible path respecting the constraints from the previous section, i.e. such that the resulting tree satisfies all our constraints. We continue like this, merging the next entry to the current tree, until all the entries are connected to the runway. (Note that even though our heuristic was not designed to optimize any particular objective function, it actually does give the lexicographic optimum for the following problem "Find the tree in which the path from the first entry is shortest, and for each *i*=2...*n*, the path from entry *i* is as short as possible given the existence of the paths from entries 1...*i*-1".) The remainder of this section gives the details of our algorithm.

Define *R* as the ray emanating from *f* in direction -*F* (i.e., *R* is aligned with the runway but goes in the opposite direction). We will make the following non-degeneracy assumption: none of the *n* entries belongs to *R* (the assumption may be made without loss of generality since it can be satisfied by an infinitesimal perturbation of the input). We place on *R* a *dummy entry* (refer to Fig. 5) at a point *o* far from *f* (at distance *nL*).

Let now *a* denote the first (real, nondummy) entry point, i.e., entry number 1. (Just as any path in the UPR tree,) the path from *a* consists from the segment *at* followed by arc *tf* of a circle *C* of (*f,F*), where *t* is the point of tangency of the line from *a* to *C*. Let *b* denote the second entry point. Let us see which constraints the first path, *atf*, creates for the routing of the second path, *b*-???-*f*.

First of all (Fig. 5), the second entry is not allowed to directly connect to any of the circles of (*f,F*) (as it would like to do in the UPR solution). Indeed, if it connects to the circle *C* at some point *s*, then either *s* or *t* will be a merge in the middle of the arc, violating the merge-at-arc-endpoint constraint (or equivalently, the segment-out-of-merge constraint). On the other hand, if *b* connects directly to the other circle *C'* of (*f,F*), then arcs of both circles of (*f,F*) will belong to the tree, which would violate the one-path-straight-at-a-merge constraint at the point *f* where the two arcs would merge.

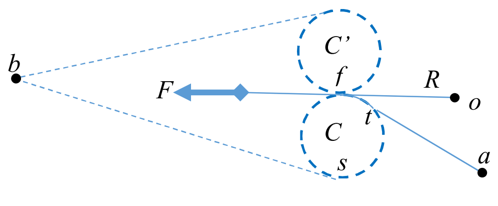

Figure 5. *b* cannot connect to *C* or *C'*.

Next, let us see how the path from *b* may reach *f*. There are 2 cases: either the path from *b* is completely disjoint from the path *atf* from *a* (other than merging at *f*), or the path from *b* connects to *atf* somewhere in the middle of the paths. Let us look closer at each case (Fig. 6):

1. If *f* serves as the merge point, then the path from *b* must reach *f* with a segment (otherwise two arcs would merge at *f*, violating our constraints). That is, the last segment of the path must belong to the ray *R*. In addition, to satisfy the merge points separation constraint, the startpoint of this last segment must lie at distance at least *L* from *f* (recall that we treat *f* as a merge). This defines a *forbidden segment ff'* on *fo*, where *f'*=shift(*f*,-*F*,*L*) is *f* shifted by *L* along *R* away from the runway, on which the merging is not allowed.
2. If the path from *b* attaches to *atf*, the attachment point must belong to the segment *at* (for otherwise the arc *tf* would contain the merge point in its interior -- a violation of our constraints). In addition, to satisfy the merge points separation constraint, the merge must lie at distance at least *L* from *t*. This defines a *forbidden segment tt'* on *ta*, where *t'*=shift(*t*,*ta*,*L*) is *t* shifted by *L* along *ta* towards *a*, on which the merging is forbidden.

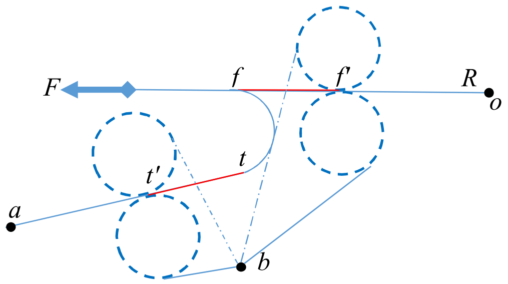

Figure 6. Forbidden segments are red. The best connection from *b* is found among the 4 possibilities, 2 of which (dashed-dotted) are excluded due to the intersections.

It follows that the best way to connect *b* to *f* is the best among (at most) 4 possibilities: connecting to the left side of the segment *of'*, to the right side of *of'*, the left side of *at'* or to the right side of *at'* (some, but obviously not all, of the 4 possibilities may be discarded since they would involve crossing between *atf* and the path from *b*). The next theorem asserts that connection to a segment is best done via the tangent to the circle touching the segment at its endpoint:

Theorem 5. Let *cd* be a segment (of length at least *L*); let *c'* be the point on *cd* at distance *L* from *c*, and let (*c',c'c*) denote the configuration whose point is *c'* and the direction is *c'c*. Let *b* be a point outside the circles of (*c,c'c*), and let *bq* be the tangent from *b* to the circle *L*(*c',c'c*) so that *qc'* follows the circle counterclockwise (Fig. 7). Then *bqc'c* is the optimal cc path that connects *b* to the left side of *cd* under the restriction that the path is not allowed to connect on *c'c*.

Proof. Let *m* be the point where the path from *b* merges to *cd*; we claim that if $m \neq c'$, then the path *bmc'* is not the shortest path. Indeed, if it were not the case, then *bmc'* would have been the shortest cc path from b to $(c',c'c)$ – a contradiction to Theorem 3 (according to the theorem, the path must be a segment followed by an arc). q.e.d.

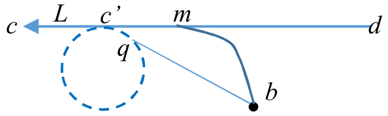

Figure 7. By Theorem 3, shortest bc' path must have the segment-arc form.

The best way to connect to the segment from the right is symmetric -- drop the tangent to the right circle of $(c',c'c)$.

Deciding the best way of connecting the entry b to the runway is easy now: Compute the *distance labels* (i.e., distances to the root), $d(t')$ and $d(f')$, of the endpoints of the forbidden segments. Then, for each of the (at most 4) connection possibilities, find the optimal connection using Theorem 4: draw the tangent from b to the corresponding circle. Finally, choose the best connection -- the one with the smallest distance label plus the segment-arc connection length.

The above extends to attaching an arbitrary entry i to the tree (Fig. 8): At any point during the algorithm's execution, maintain the set of forbidden segments and distance labels for their endpoints; the segments are the potential connection portals for the entry. Choose the best connection by drawing tangents from i to the circles of the configurations defined by the forbidden segments. The segment cc' to which i is connected (via a segment-arc connection idc') stops being forbidden. The new connection defines a new forbidden segment dd' on the ray di (plus a new forbidden segment on R, in case the path from i uses R), and the algorithm proceeds to connecting the next entry, i+1.

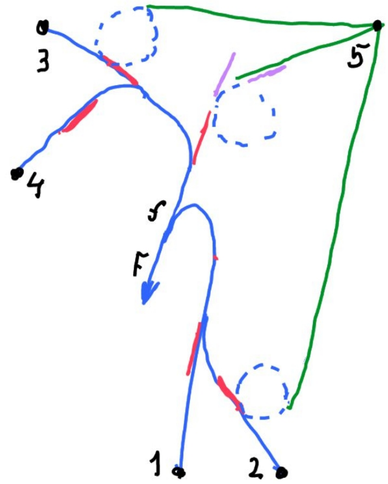

Figure 8. Entry i=5 is connected via a tangent circle (feasible circles shown dasahed) of the configuration defined by one of the forbidden segments (red) in the current tree (blue); in this instance, the connection via R is the shortest out of the three green intersection-free connections. After entry i is added to the tree, the forbidden segment on R will be removed from the set of forbidden segments, but 2 new segments (shown pink) will become forbidden. (The merge incident to the new entry always recieves a new a forbidden segment near its tangency to the arc: in addition, if the new connection uses R, a new forbidden segment is installed on R, since R must always feature a forbidden segment).

VI. Results

We implemented our algorithm for Arlanda airport in Stockholm terminal airspace (Fig. 9); the output routes are smoother and shorter than the current STARs [20], but cross through noise restricted areas (there is no obstacle avoidance is in our model). Below, we give some arguments in support of our choices for the implementation parameters values.

Choosing *r*=5-7nmi. FAA Orders 8260.54A and 8260.58 rule that a turn must have radius of least 3nmi. For an upper bound on r, we consulted Garmin's RF report [21] in support of FAA Memorandum of Agreement # DTFAWA-11-A-80009, which established 13.034nmi as the largest radius of RF legs. The report also limits the radius from below by half-width of the obstacle evaluation area (OEA), and FAA Order 8260.52 1.7 defines OEA as 2×RNP on either side of the segment centerline. Based on the above considerations, we feel that r=5-7nmi would be an appropriate choice. This was confirmed via personal communication with the LFV and operatives in Arlanda airport.

Choosing *L*~10nmi. FAA Order 8260.58 Vol. 6 gives criteria for the design of arrival routes RF legs (section 2.5.3 of Order 8260.54A states very similar criteria); it rules that RF legs are not applicable to the final segment, and that in the intermediate segment the legs must terminate farther than 2nmi from the final approach fix (FAF). ICAO instrument approach charts (available e.g., from Jeppesen) show Arlanda's FAF at approximately 7nmi from the threshold, which gives L~10nmi for the length of the straight segment before the touchdown. Using the same L=10nmi for merge points separation gives ATCOs a couple of minutes between the merges (assuming speed of 200-300knots) -- normally sufficient to communicate possible contingencies. Overall we believe that L equal to 1.5-2 times r looks reasonable, which is being confirmed with the industry experts.

VII. Conclusion and Discussion

We presented a greedy (iterative) algorithmic approach to design of STARs with RF legs. Our STAR provides the shortest path from the most loaded entry, then the shortest available path from the second most loaded entry, and so on. The priority of the entries (i.e. the order in which they are added to the tree via the shortest paths) may be changed by the user; for instance, if a new entry is added to the TMA, the shortest path from it may be added to the tree irrespectively of the entry's load (e.g., to pick up the increasing traffic from East, a new entry was established some time ago in Stockholm TMA; the route from the entry was designed so as not to disturb the routes from the other entries, to minimize ATCOs adaptation period).

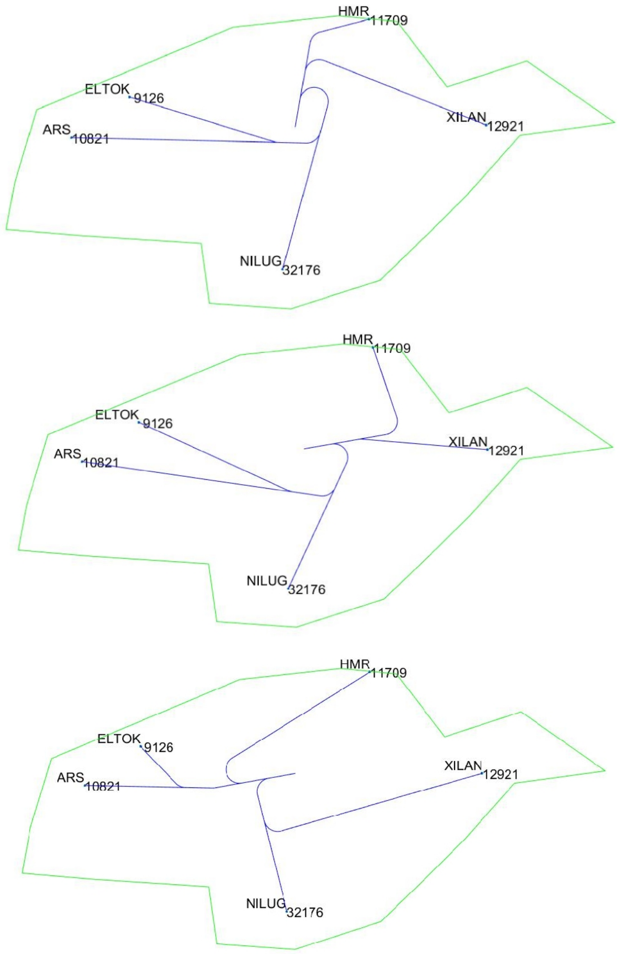

Figure 9. Our cc trees for the 5 entries (whose loads are shown next to them) to Stockholm TMA (whose boundary is shown green). Top to bottom: RWYs 19L, 26 and 08 in Arlanda.

The presented method has consequences also outside our setting in which the locations of the entry points are fixed, and gives insights, for instance, into airspace *redesign* initiatives in which the entries may be moved further into the enroute portion (Advanced AMAN [22] in an extended terminal airspace, resolving potential in-TMA conflicts further away from the RWY, may be viewed as one step towards implementation of 4DT time-based operations). An immediate question that arises is "What would be the advantages of moving an entry point, and how far should it move?" Our algorithm provides an answer and a way to quantify benefit of moving the point: if the entry is too close, the cc routes might not fit into our cc STAR, and hence longer routes would have to be used; on the other hand, it is easy to see that the entries lying further than nL from the RWY can always be connected via our cc tree (e.g., as a "caterpillar" graph with all entries merging to the ray R; see Fig. 10, left).

Our approach can handle more constraints. For instance, it can be seen that our output sometimes includes very short arcs. Even though we did not find any lower bound on the arc length in the RNP specifications, we suspect that it may be hard for the FMS to go into RF mode for a brief period and then go back. Short arcs may be forbidden within our algorithm simply by enlarging the forbidden segment until the tangent from the leaf touches the tangent circle far enough along the circle from the merge point (Fig. 10, right).

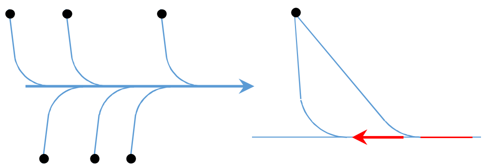

Figure 10. Left: A feasible way to connect all entries is by merging them all to R. Right: Enlarging the forbidden segment lengthens the connecting arc.

Further possible extensions may include taking into account the Earth curvature and 3D, as well as interaction with sector boundaries, e.g., placing merges deep inside sectors, bounding the number of merges per sector, etc. Another important research direction is computing SIDs in addition to STARs (in this paper we ignored the departing traffic completely). These questions are outside the scope of this paper, but may be investigated in future work.

We could vary the separation minimum L depending on the distance from the runway: since aircraft fly with higher speed on the initial approach, the ATCOs might want to have larger separation distance between the merges on the initial parts of the paths. Also, the separation on the initial approach may be different for different entries because they may serve different fleet types (in general, determining the fleet mix is an important step in airspace design [16]). Still, we believe that having only one parameter in the model (apart from the radius r – but this one cannot really be viewed as user-tunable, rather, it is determined by the aircraft equipage) has its advantages, as it makes the model easier to understand and interact with. In support of the claim, we coded a simple GeoGebra applet allowing the user to change L and see how the tree changes (the topology of the tree remains fixed); Fig. 11 presents a couple of the applet snapshots (we put the full interactive applet at https://tube.geogebra.org/m/i2I0x2Fn). Last but not least, it is of interest to investigate the robustness of our trees: the full solution would be to split the L-axis into intervals within which the topology of the tree remains the same – this, in particular, will identify *robust topologies* whose associated intervals are long; good values for L are those in the middle of the intervals, since using such values allows to change L without changing the tree topology (decreasing L may be suggested by aircraft in order to land sooner, while increasing L may be requested by ATCOs in order to get more time for control, e.g., in inclement weather or high traffic volume).

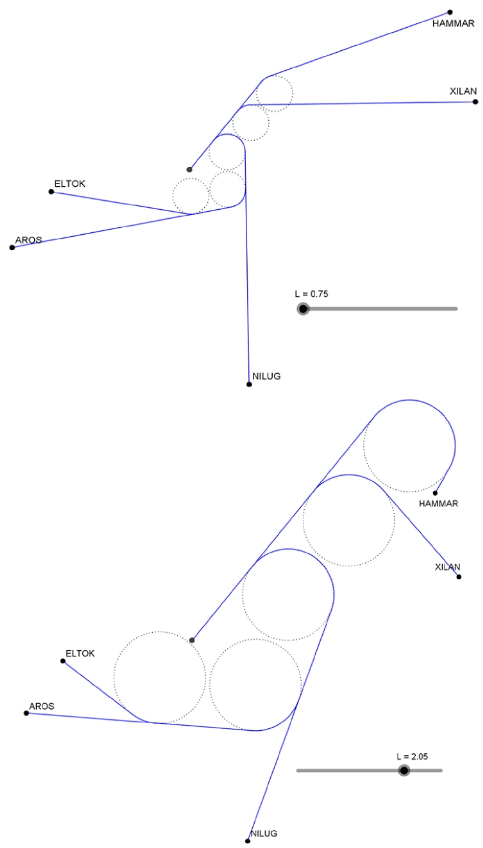

Figure 11. Snapshots of the applet for a small and a large L.

ACKNOWLEDGMENT

I thank anonymous reviewers for their helpful comments, the reference group of ODESTA project---Patrik Bergviken (Prod TWR/TMC TMC Landvetter, LFV), Robert Graham (EUROCONTROL), Johan Holmer (Trafikverket, the Swedish Traffic Agency), Anders Ledin (Swedavia), Anne-Marie Ragnarsson (Transportstyrelsen, the Swedish Transportation Authority)---for valuable discussions on the objective function, and Pat Manzi (ATCC Arlanda) and Kaija Sterner (LFV) for sharing their domain expertise during the fruitful meetings.

REFERENCES

[1] FAA. Performance based flight systems branch

[2] FAA. NextGen implementation plan

[3] ICAO. Performance-based navigation Manual (Doc 9613)

[4] EUROCONTROL. European airspace concept handbook for PBN implementation

[5] Network Manager, EUROCONTROL. Introducing performance based navigation (PBN) and Advanced RNP (A-RNP)

[6] A. Markov. Some examples of the solution of a special kind of problem on greatest and least quantities. Soobshch. Kharkovsk. Mat. Obshch. 1, 250–276 (1887). In Russian

[7] L. Dubins. On curves of minimal length with a constraint on average curvature. Am. J. Math. 79, 497–516 (1957)

[8] H. Sussmann, G. Tang. Shortest paths for the Reeds-Shepp car. Research report SYCON-91-10, Rutgers University, New Brunswick, NJ (1991)

[9] J.-D. Boissonnat, A. Cerezo, J. Leblond. Shortest paths of bounded curvature in the plane. Int. J. Intell. Syst. 10, 1–16 (1994)

[10] K. Buchin, B. Speckmann, K. Verbeek: Flow map layout via spiral trees. IEEE Trans. Vis. Comp. Graph. 17(12): 2536-2544 (2011)

[11] K. Buchin, B. Speckmann, K. Verbeek: Angle-restricted Steiner arborescences for flow map layout. Algorithmica 72: 656-685 (2015)

[12] J. Krozel, C. Lee, J. Mitchell. Turn-constrained route planning for avoiding hazardous weather. ATCQ. 2006

[13] D. Michalek, H. Balakrishnan. Identification of Robust Terminal-area Routes in Convective Weather, Transp Sci 46(1)56-73, 2012

[14] D. Michalek, H. Balakrishnan: Dynamic reconfiguration of terminal airspace during convective weather. IEEE CDC 2010: 4875-4881

[15] J. Krozel, J. Prete, J. Mitchell, J. Kim, J. Zou. Capacity estimation for Super-Dense Operations. AIAA GNC 08

[16] J. Prete, J. Krozel, J. Mitchell, J. Kim, J. Zou. Flexible, performance-based route planning for Super-Dense Operations. AIAA GNC 08

[17] S Choi, JE Robinson III, DG Mulfinger, BJ Capozzi. Design of an optimal route structure using heuristics-based stochastic schedulers. DASC '10

[18] X.-N. Bui, J.-D. Boissonnat. Accessibility region for a car that only moves forwards along optimal paths. [Research Report] RR-2181, 1994

[19] XN Bui, JD Boissonnat, P Soueres, JP Laumond. Shortest path synthesis for Dubins non-holonomic robot, ICRA'94

[20] LFV. AROWeb. www.aro.lfv.se/Editorial/View/IAIP?folderId=56

[21] Garmin. Doc 005-00586-21: Radius-to-Fix leg project report (2013)

[22] F. Dijkstra, D. Mijatovic, R. Mead. Design options for advanced arrival management in the SESAR context. ATCQ. 2011