Using DIRECT to Solve an Aircraft Routing Problem *

M.C. Bartholomew-Biggs , S.C. Parkhurst and S.P. Wilson
Numerical Optimisation Centre, University of Hertfordshire

ABSTRACT
In this paper we discuss a global optimization problem arising in the calculation of aircraft flight paths. Since gradient information for this problem may not be readily available, a direct-search algorithm (DIRECT), proposed by Jones et al. [11] appears to be a promising solution technique. We describe some numerical experience in which DIRECT is used in several different ways to solve a sample problem.

1 Introduction

The *Aircraft Routing Problem* involves finding an "optimum" flight path between various obstacles from a given origin to a given destination. Obstacles would clearly include geographical features, but might also be more general "no-fly zones", used, for instance, to separate incoming and outgoing traffic near an airport. In military terms a no-fly zone might surround a threat, such as an enemy radar or missile site. In practice, the routing problem will usually also include constraints on rendezvous time at the destination (and perhaps at other points on the path); and obviously any solution must allow for speed and manouevrability limits on the type of aircraft being used. Many of these aspects are probably similar to those found in steering problems for other kinds of vehicle or in the planning of paths for cables or pipelines. More distinctive features of aircraft routing include "one-way" regions where flight must be in a particular direction. In the military context, a route must also be assessed in terms of "visibility" (it being desirable to exploit the terrain to hide the aircraft as much as possible). One way to estimate such visibility would be to sample the terrain along a number of (possibly randomly chosen) directions from each point along the aircraft path in order to determine the shortest distance to high ground that might ofer concealment. In its ultimate form, the routing problem

*This research is supported by BAE SYSTEMS, Rochester, England

would be posed for multiple aircraft, possibly of various types and flying different missions, in the same geographical area.

More discussion about practical aspects of aircraft routing can be found in [9] and [10] which also describe a heuristic routing algorithm. The present paper is concerned with the application of classical optimization techniques to aircraft routing; and in this preliminary study we confine ourselves to a relatively simple two-dimensional representation of the problem. This is used to provide a case-study in the use of an ingenious direct-search global optimization method proposed by Jones et al. [11]. This method, described more fully in a later section, uses Lipschitz constant arguments to motivate a deterministic search strategy which has been shown elsewhere [14] to be more effective than random-search techniques on relevant examples.

2 A basic 2D representation

We shall consider the aircraft routing problem in two dimensions – i.e., we shall find the ground-plan (flat earth) of a route which avoids a number of no-fly zones that we shall henceforth call "threats". A route is defined by given start and end points and a number of intermediate *waypoints*. The co-ordinates of these waypoints are the optimization variables; and we shall assume that the flight path follows straight lines between waypoints. To characterize optimality of a route, we suppose that we want the distance flown to be as short as possible, subject to suitable avoidance of the threats. Therefore, for any choice of waypoints, we must calculate first the Euclidian length of the corresponding route, denoted by L, say. We then determine whether the route passes through any of the threats; and, if so, we calculate the length of path lying inside each one. If the route lies within the i-th threat for a distance L_i, say, then the "cost" of the route can be expressed as a composite function such as

$$C = L + \sum_i \rho_i L_i^p \tag{2.1}$$

where ρ_i is a *penalty parameter* associated with the i-th threat and p is an integer exponent to be discussed below. Our aim will be to choose the waypoints so as to minimize C. The balance between reducing the flight path length and respecting the threats will depend on the choice of the parameters ρ_i.

Suppose there are n waypoints $w_j = (w_{j1}, w_{j2})$ for $j = 1, .., n$. For consistency we shall denote the starting point as w_0 and the destination as w_{n+1}. We can then consider the whole route in terms of "legs" from w_{j-1} to w_j for $j = 1, ..., n+1$. We let u_j denote the vector $w_j - w_{j-1}$. Hence a typical leg contributes

$$l_j = \sqrt{(u_{j1}^2 + u_{j2}^2)}$$

to the total length L. We now need to determine whether leg j passes through any of the threats.

In the test examples discussed in this paper we shall, for convenience, use circular threats; but in general we shall need to deal with no-fly zones and geographical obstacles which have irregular boundaries. Therefore we must calculate threat violations using a sampling process along each leg of a route. We suppose that it is possible to determine whether any point $(x,\ y)$ is inside or outside a threat but that no explicit expression is available for the threat boundary. (For example, if a threat is simply an area of high ground then a geographical database which provides terrain height for given longitude and latitude will enable us to determine whether some constant altitude flight-path is feasible or not.)

In the algorithm below we use linear interpolation to estimate the points at which leg j intersects threat i. In order to implement the algorithm we can consider σ_{max} to be a maximum step size to be used in sampling a leg of a route. This will imply that the number of sampling points along leg j can be taken as $K_{max} = \lceil l_j/\sigma_{max} \rceil$. We let $\delta\lambda = 1/K_{max}$ and suppose that u_k denotes a sampled point in leg j, i.e.

$$u_k = w_{j-1} + k\delta\lambda(w_j - w_{j-1})$$

where k goes from 0 to K_{max}. Suppose that T_i is a function of position such that $T_i(u_k) \leq 0$ indicates that u_k is inside threat i while $T_i(u_k) > 0$ confirms that u_k is outside. Then an algorithm for calculating the "in threat" length l_{ji} is as follows.

Compute l_j as length of leg j. Set $l_{ji} = 0$
Set $k = 0$, $K_{max} = (\delta\lambda)^{-1}$
If $T_i(u_0) \leq 0$ then $\lambda_b = 0$
For $k = 1, .., K_{max}$
  If $T_i(u_k) \leq 0$ and $T_i(u_{k-1}) > 0$ then
    set $\kappa = T_i(u_k)/(T_i(u_k) - T_i(u_{k-1}))$, $\lambda_b = (k - \kappa)\delta\lambda$
  If $T_i(u_k) > 0$ and $T_i(u_{k-1}) \leq 0$ then
    set $\kappa = T_i(u_k)/(T_i(u_k) - T_i(u_{k-1}))$, $\lambda_e = (k - \kappa)\delta\lambda$
    set $l_{ji} = l_{ji} + (\lambda_e - \lambda_b)l_j$
If $T_i(u_{Kmax}) \leq 0$ then
  set $l_{ji} = l_{ji} + (1 - \lambda_b)l_j$

In what follows we use a version of the function (2.1) which includes two extra features to make our examples a little more realistic. We do not permit routes which involve sharp turning manoeuvres; and furthermore we do not allow waypoints to be too close together. Therefore our cost function includes penalty terms connected with these quantities. The angles ϕ_j between successive legs are given by

$$\phi_j = \cos^{-1}\{\frac{u_j^T u_{j+1}}{||u_j||||u_{j+1}||}\}$$

If ϕ_{max} is the limiting turn angle and l_{min} denotes the least acceptable leg-length

then we can extend the cost function definition (2.1) as

$$C = \sum_{j=1}^{n+1} (l_j + \mu((l_{min} - l_j)_+)^2 + \sum_{i=1}^{m} \rho_i l_{ji}^p) + \sum_{j=1}^{n} \nu((\phi_j - \phi_{max})_+)^2 \qquad (2.2)$$

where μ and ν are positive penalty parameters. The penalty terms involving the bounds on turn angle and leg length are standard quadratic loss functions. However, the choice of the exponent in the penalty term for threat violations is discussed in [14], using an argument based on the *analytical* calculation of l_{ji} for circular threats. It can be shown that, provided $p \geq 3$, there is continuity in the first derivatives of C in the limiting case when changes in waypoint position cause a leg to enter or leave the i-th threat. This is essentially because the in-threat leg-length calculation for l_{ji} involves a square root. Numerical experience in [14] also suggests that it is preferable to use $p = 3$ rather than $p = 4$ in (2.2). We assume that these observations carry over to more general situations where threats have irregular boundaries.

We note that the sampling and linear interpolation method of calculating the l_{ji} means that it will be difficult to evaluate first derivatives of C with respect to the waypoint co-ordinates. Hence it might be appropriate to consider gradient-free methods for minimizing (2.2). (This remark has more force if we take account of the "visibility" constraints, mentioned in the introduction, which appear in more realistic versions of the routing problem. These constraints are typically computed on the basis of random probes from the current aircraft position to locate the nearest regions of high ground and such computations seem unlikely to be differentiable.) Furthermore we need a global minimization method, since experience reported in [14] shows that the function (2.2) may have several local minima. This is easy to visualise if we think of a problem with just one circular threat and one waypoint. Typically if the "best" route passes the threat on the left (say) we can expect there to be another, at least locally, "best" route which skirts it on the right. Waypoint positions between these two solutions will cause the route to pass through the threat and hence incur high values of C. Thus we have the classic case of two regions with low function value separated by a ridge with larger values of C.

DIRECT is a gradient-free global minimization method proposed by Jones et al. [11]. It is a deterministic approach which systematically searches within an initial "hyperbox" in the space of the variables, exploration being focussed on regions which are judged to be "potentially optimal" on the basis of tests involving Lipschitz constants. This approach is outlined in the next section

3 An outline of DIRECT

This section provides a general description of DIRECT, giving sufficient detail to support our later discussion of different ways in which it can be applied to the

route-finding problem. A fuller discussion, including implementation issues, can be found in [11].

DIRECT searches for the minimum of a function $f(x)$ within a hyperbox defined by $l_i \leq x_i \leq u_i$ for $i = 1, ..., n$. It proceeds by systematic subdivision of this initial region into smaller hyperboxes. The distinctive feature of the algorithm is the method of choosing which boxes to subdivide on each iteration.

At the start of a typical iteration of DIRECT, the initial exploration region will be completely covered by (say) K hyperboxes of differing sizes. Specifically, the size of each box is represented by $\delta_i,\ i = 1, ..., K$ the distance from the centre to a corner. Function values $f_i,\ i = 1, .., K$ are known at the centre of each box. Box i is regarded as *potentially optimal* if there exists a Lipschitz constant L such that both

$$f_i - L\delta_i < f_j - L\delta_j \text{ for } j = 1, .., i-1, i+i, ..K \tag{3.1}$$

and

$$f_i - L\delta_i < f_{min} - \varepsilon |f_{min}| \tag{3.2}$$

(where f_{min} is the least value of f found so far and ε is a small, positive, user-supplied constant). Condition (3.1) indicates that it is *possible* for a point in box i to have a smaller function value than a point in any other box. Condition (3.2) ensures that points in box i also have the possibility of making a non-trivial improvement on the best point found so far.

Any hyperbox which is identified as potentially optimal is then subdivided. The simplest way of doing this involves splitting its longest side into three parts and hence produces two new function evaluations. (More complicated rules for subdivision are described in [11] for hyperboxes with more than one longest side.)

It is worth noting that, in practice, the number of potentially optimal boxes on a particular iteration is usually quite a small fraction of K. Hence the subdivision costs are typically not excessive. Furthermore, the process of determining whether box i satisfies conditions (3.1) and (3.2) need not be as computationally expensive process as might at first appear, since [11] explains that it is not necessary to perform the test (3.1) explicitly for all $j \neq i$. If the number of different box sizes is $\hat{K}$ (usually $<< K$) then we only need to make comparisons between the $\hat{K}$ boxes which have the smallest centre-values for their size.

Computational experience with DIRECT has shown it to be quite efficient. In the numerical tests reported in [11], DIRECT outperforms several other direct search optimization methods in terms of numbers of function values needed to find global solutions to some standard test problems. Preliminary experience of applying DIRECT to the route-finding problem in [14] shows that it minimizes (2.2) more efficiently than than the methods in [1] and [5] which employ random searching. The use of DIRECT in aerospace design problems (with quite large numbers of variables) is reported in [2], [3], [4], [6], [7], [12] and [13]. In [7], in particular, it is noted that it does better than a trajectory-following approach on functions

with deep and widely spaced minima. This remark is relevant to the route finding problem because the various locally optimum routes are likely to be clearly distinguishable by virtue of passing on different sides of at least one of the threats.

The value of the parameter ε in (3.2) is a user-choice to be made in setting up DIRECT. It is in fact almost the only such choice – which means that it is quite an easy algorithm to use. The other parameter that has to be selected is the number of iterations to be performed. Although the authors provide a proof of ultimate convergence of DIRECT in [11], they do not suggest a practical stopping rule for the algorithm and merely suggest that a fixed number of iterations be done. This strategy can lead to uneccessary work being done in some cases, while on other problems the global solution may not be found. It might be somewhat better – though certainly not an infallible test – to continue until no significant improvement in f has been found for (say) fifty iterations.

This question of a stopping rule is one of the issues we explore in the next section where we consider a number of different ways of applying DIRECT to the aircraft routing problem.

4 The routing problem - a case study

In this section we shall use some experiments with DIRECT to illustrate some important issues in solving the aircraft routing problem. These issues include the initial choice of waypoints – both their number and their guessed position. Once we have decided how many waypoints to use, a possible "automatic" strategy might be to place these at equal intervals along the straight line joining the departure and destination points. This could provide a useful initial bias towards the distance minimization component of the cost function, leaving it subsequently to the penalty terms to steer the aircraft around the threats in an efficient manner. It is also important to consider the choice of values for the penalty parameters in (2.2), since it is well-known that minimization becomes numerically difficult if these are chosen "too large". We seek to shed some light on these matters by describing several approaches to a test problem which involves the following threats

centre(km)	radius(km)
75,29	9
80,66	15
95,115	22.5
135,131	20
140,67.5	37.5
197,125	30

We shall consider the calculation of a route from (50, 30) to (167, 107) and back, imposing a minimum stage length of 10 km and a maximum turning angle of 42.5^o

(which ensures that the return journey cannot simply retrace the outward one).

4.1 Approach 1 - separate calculation of outward and return legs

We first consider the application of DIRECT to the outward journey. Let us *assume* that two waypoints will be sufficient and as an initial guess we shall place them both near the midpoint of the line joining the departure point and destination. This is obviously a very poor starting estimate. Specifically we use

$$(108 \pm 60, 68 \pm 40), \quad (109 \pm 60, 69 \pm 40) \tag{4.1}$$

which allows the optimization procedure to search throughout a rectangle slightly larger than the one with corners at the departure point and destination.

If we use penalty parameters $\mu = \nu = \rho_i = 1$ in (2.2) and do 64 iterations of DIRECT then the best route found has waypoints at (68,77) and (125,105) with a length of 155.6 km (and no threat violation). The maximum turn angle is between stage one and stage two and is 43.2^o. This result costs 1343 function calls.

If we let DIRECT run for 128 iterations (3917 function calls) from starting guess (4.1) then the maximum turn angle is reduced to 43^o at the expense of a slight increase in route length and a threat violation of about 0.14 km. However we can also operate DIRECT in a *restart mode*, whereby we run it again from an initial guess centred upon the best point found in the first 64 steps. Thus, if we restart from

$$(68 \pm 60, 77 \pm 40) \quad (125 \pm 60, 105 \pm 40) \tag{4.2}$$

and do a further 64 iterations (1845 function calls) we get a feasible route of length 155.6 km with no threat violations and a maximum turning angle of 43^o. This is slightly better than we got by 128 iterations from the original guess (4.1) and the cost is 3198 function calls as opposed to 3917.

Since DIRECT is a global optimization method, a change in starting point should not influence the solution it ultimately obtains. The benefit of a restart lies in the possibility that it will cause DIRECT to approach the global solution more quickly and cheaply. As explained earlier, each iteration of DIRECT examines the current set of hyperboxes to determine which are potentially optimal. The number of boxes to be examined (and probably the number of boxes to be subdivided) gets larger as the number of iterations increases. After a restart based upon the current best point, the iterations of DIRECT will be cheaper than they would have been without the restart. Moreover, because DIRECT will now have one box which is centred on a "good" function value, it may be that fewer boxes will be found to be potentially optimal in the second and subsequent runs of DIRECT. Certainly the experience of this first example is that two lots of "cheap" iterations can be better value than an equivalent number of increasingly expensive ones. (Note that in restarting DIRECT from (4.2) we have chosen to keep the initial hyperboxes the same size as at the

start. A case could be made for making them smaller and we shall return to this idea later on.)

Since the turning angle constraint has not yet been satisfied, a possible strategy would be to increase the parameter ν in (2.2). However it may be easier just to insert an extra waypoint and solve the problem again. This remark is well worth making because further experiments (not reported in detail here) suggest that it is not possible to obtain a feasible route with just two waypoints. Specifically, when ν is increased, DIRECT is able to find approximate minima of (2.2) which involve smaller turning angles; but this happens only at the expense of threat violations. This illustrates the fact that is quite easy to pose routing problems which have no feasible solution. It is worth mentioning, however, that DIRECT can be used to check whether a problem has a feasible point. This is done by treating the maximum constraint violation as the objective function and finding whether or not it has a global minimum of zero.

In order to find a 3-waypoint route from (50,30) to (167, 107) we use the previous best result to provide the starting guess

$$(68 \pm 5, 77 \pm 5) \quad (97 \pm 30, 91 \pm 14) \quad (125 \pm 5, 105 \pm 5) \tag{4.3}$$

Notice that we have put fairly small boxes around the two "known" points and placed an extra point midway between them. The solution we now get has an overall length of 154.9 km. It is obtained in 4622 function calls with the 2×64 restart version of DIRECT and the waypoints are

$$(67.4, 75.1) \quad (76, 82) \quad (131.4, 104.5) \tag{4.4}$$

In this case a very similar solution can be obtained in 128 iterations and 4609 function calls without a restart.

We now consider the return part of the route. In order to prevent there being too tight a turning angle at (167, 107) we include the last outward leg in our calculation and seek a route from (131.4, 104.5) to (50, 30), forcing it to pass through (167,107) by enclosing this waypoint inside a very small box. Thus we take the starting guess for DIRECT as

$$\begin{aligned} &(167 \pm 0.1, 107 \pm 0.1) \quad (109 \pm 100, 67 \pm 50) \quad (108 \pm 100, 66 \pm 50) \\ &(107 \pm 100, 65 \pm 50) \quad (106 \pm 100, 64 \pm 50) \end{aligned} \tag{4.5}$$

Here we have *assumed* five waypoints will be sufficient; and the unknown waypoints have been placed near the midpoint of the line from (167, 107) to (50, 30). Note that the boxes enclosing these waypoints are substantially bigger than would be necessary to allow them to move within the rectangle with the starting point and destination at its corners. Experience has shown that we can miss the global optimum if we do not give the waypoints enough freedom.

For the return route problem we have used $\nu = 10$ in (2.2). This increase in penalty parameter associated with the turning angle limit is motivated by experience: the homeward flight path needs to turn through a total angle of about 120^o, and the waypoints which minimize (2.2) imply some unacceptably sharp turn angles if we retain the previous choice $\nu = 1$. We are of course free to use different penalty parameter values in the outward and return calculations because, at present, we are effectively treating them as separate problems. Our purpose in mentioning the need to increase ν is to point out that the appropriate choice of penalty parameters in (2.2) is problem-dependent.

Table 1 shows details of the best points found after different numbers of DIRECT iterations starting from (4.5).

itns	fns	route length	threat violation	max turn angle
128	7141	244.7	0	43.6
256	16531	242.7	0	43.0
512	36619	242.7	0	43.0

Table 1: **Return routes from DIRECT with initial guess (4.5)**

Table 2 shows that the restart mode of DIRECT can obtain a good feasible route which satisfies the turn angle constraint in fewer iterations and function calls than those quoted in Table 1. In the best return route, corresponding to the last row of

itns	fns	route length	threat violation	max turn angle
2×64	6602	252.6	0	42.5
4×64	15552	248.2	0	< 42.5
6×64	24574	242.4	0	42.5

Table 2: **Return routes from DIRECT using restart and initial guess (4.5)**

Table 2, the calculated waypoints between (167, 107) and (50,30) are

$$(177.5, 98.6)\ \ (183.5, 65.2)\ \ (161, 30)\ \ (93, 11.3) \tag{4.6}$$

Putting together the routes defined by (4.4) and (4.6) we get a total route length of 361.6 km (recalling that the calculated outward and return routes have covered the stage from (131.4, 104.5) to (167, 107) twice).

4.2 Approach 2 - simultaneous calculation of outward and return legs

The approach we have used in the previous section is unlikely to yield a global optimum for the full round-trip problem, although it should provide a useful first approximation. We can seek a further refinement of the route found in the previous subsection by running DIRECT on the whole outward and return problem with

a starting guess given by (4.4) and (4.6) and using a margin of ± 5 km on each waypoint coordinate. In this case we use $\nu = 10$ for for turning angles on both the outward and return stages. After 128 iterations (15961 function calls), small adjustments have been made to the waypoints which reduce the route length to 358.8 km. This seems to be near-optimal, since no significant improvement is obtained by allowing DIRECT to run for 256 iterations.

We could, of course, have attempted the whole routing problem in one go, without obtaining the partial solutions (4.4) and (4.6) first. We now perform this calculation (*assuming* we already know that a total of eight waypoints is sufficient). We take (50, 30) as both departure point and destination and include (167, 107) as a waypoint with effectively zero tolerance on the components. If we follow the precedent of starting guess (4.5) and put the other guessed waypoints near the midpoint of the line between (50, 30) and (167, 107) with margins $\pm 100, \pm 50$ on the coordinates then we get results from DIRECT as shown in Table 3.

itns	fns	route length	threat violation	max angle	min leg
128	7927	356.9	55.8	44.7	1.4
256	20423	362.4	17.9	42.8	9.0
384	35851	368.9	16.9	< 42.5	> 10
512	49963	368.9	16.5	< 42.5	> 10

Table 3: **Routes from DIRECT on the out and return problem**

It is clear that DIRECT has some difficulty in avoiding the threats. We therefore try the algorithm in restart mode, this time allowing 128 iterations in each cycle. Results appear in Table 4 and these are considerably better than what we obtained without restarts. The threat violations are much reduced and the best route length is a little less than was obtained when starting from the initial guess given by (4.4), (4.6). Nevertheless, these tables (and other numerical tests not reported in detail) suggest that DIRECT experiences more difficulty and uses more effort to solve the full out-and-home routing problem "from scratch" than it does when the problem is considered in two separate phases.

itns	fns	route length	threat violation	max angle	min leg
2×128	23726	366.8	0.2	< 42.5	> 10
3×128	41913	359.6	0.3	42.5	> 10
4×128	59936	357.7	0	< 42.5	> 10

Table 4: **Routes from DIRECT using restart on the out and return problem**

4.3 Approach 3 - building a complete route by adding waypoints

We can use experience from the preceding subsections to automate the process of building a route, including the question of whether to add extra waypoints. The

approach outlined below is partly motivated by the encouraging performance of restart mode DIRECT.

We begin a route-finding solution by performing M iterations of DIRECT on the full out-and-return problem, but using only a small number of waypoints. (These can be placed initially at some convenient point such as the midpoint of the route, although it is important that a large enough box is defined to allow movement anywhere within the geographical region of interest.) The route obtained by DIRECT is then examined, and an extra waypoint is added at the midpoint of any leg which passes through threats. The enclosing box for this new waypoint is made large enough to allow movement between the "old" ones on either side of it. DIRECT is then allowed to run for a further cycle of M iterations. The "old" waypoints continue to be optimization variables for the next cycle of DIRECT, but the box surrounding them is reduced by a factor κ in each coordinate direction. A similar addition of waypoints is permitted at the next stopping point from DIRECT – although we note that waypoints are *not* added within legs whose length is less than $2l_{min}$. This process continues until two successive cycles of DIRECT produce essentially the same value of the objective function.

We now show how this approach works on the complete out-and-home problem introduced above. We begin with a three waypoint guessed solution

$$(109 \pm 100, 67 \pm 50); \quad (167, 107); \quad (109 \pm 100, 67 \pm 50) \tag{4.7}$$

and the following table shows progress when 64 DIRECT iterations are allowed in each cycle and the box scaling factor κ is taken as 0.7.

itns	fns	route length	threat violation	max angle	min leg
64	1335	373.5	62.5	107.1	> 10
128	2648	370.2	21.8	47.9	> 10
192	3953	370.9	9.6	43.2	> 10
256	5860	368.3	0	42.7	10
320	7537	363.5	0.6	< 42.5	10
384	9276	363.8	0.1	< 42.5	10

Table 5: **DIRECT with restart ($\kappa = 0.7$) and adding waypoints**

We see, by comparing Tables 4 and 5, that this approach has yielded a good solution which is comparable with what could be obtained using between two and three cycles of DIRECT in restart mode with a constant number of waypoints. In fact the number of waypoints introduced by the strategy outlined in this section is twelve, which is somewhat more than was shown to be strictly necessary using previous approaches. The numbers of function calls quoted in Table 5 are cumulative; and these figures are very much less than the numbers appearing in Table 4. Moreover the run-time for 3×128 iterations of DIRECT in Table 4 is about 70 seconds whereas the solution quoted in Table 5 only took 12 seconds.

We can obtain a better solution at somewhat greater cost if we allow 96 DIRECT iterations per cycle. In this case a feasible route of length 361.1 km (involving 11 waypoints) is found in 21 seconds. If we increase the number of iterations in each cycle to 128 then we obtain much the same solution in a time of 35 seconds.

We now consider the factor κ used to shrink the boxes around existing waypoints. The results quoted so far have been obtained with $\kappa = 0.7$. If we reduce κ to 0.5 and use 96 DIRECT iterations per cycle we are able to obtain a feasible route with length 354.2 km in only 17 seconds. Somewhat surprisingly, if we take only 64 iterations per cycle we get the results shown in the table below, which yields the best route that has been found for the example problem by any of the approaches. This calculation took just 11 seconds.

itns	fns	length	threat violation	max angle	min leg	waypoints
64	1335	373.5	62.5	107.1	> 10	3
128	2327	369.2	17.6	53.7	> 10	7
192	3781	369.7	3.0	52.7	> 10	8
256	5092	361.8	0	< 42.5	8.6	9
320	6853	353.7	0.3	< 42.5	10	9
384	8198	352.8	0.3	< 42.5	9.9	9
448	9461	352.8	0	< 42.5	> 10	9

Table 6: **DIRECT with restart ($\kappa = 0.5$) and adding waypoints**

5 Discussion and further work

Our numerical experience with DIRECT on the routing problem has been quite encouraging. In particular the use of restarts and the facility for adding extra waypoints seem to enhance the performance of the algorithm. Periodic restarts enable us to preserve the current best solution while avoiding some of the costs associated with increasing numbers of hyperboxes. It should be noted that there are other ways, more in the spirit of DIRECT, to reduce work per iteration. The source paper [11] points out that the user-selected parameter ε in (3.2) can be used to control the balance between rapid local convergence and global exploration. More recent implementations (e.g. [8]) have considered ways of automatically adjusting this balance in order to improve the efficiency of the algorithm.

We acknowledge that, at present, the promising Approach 3 can only be justified as a pragmatic solution to the problem of knowing in advance how many waypoints will be sufficient. One difficulty in establishing its convergence to a global solution to the original problem is that (even if the boxes are not shrunk) the shape of the feasible region is changed after every re-start. This *might* turn out be an advantage if the original feasible region had been drawn "too small"; but it does make it harder

to analyse the behaviour of the algorithm. It is also true that the incautious shrinking of the boxes could prevent the approach from getting to the required global solution. It is probably worth observing however that, from an operational point of view, a method which provides good feasible routes quickly may be preferable to one which locates a precise global optimum but requires more computing effort.

The example reported in the previous section now needs to be followed up by more systematic testing on a wider range of problems to establish good general values for M and κ. Consideration of M, the number of iterations of DIRECT to be performed between restarts, will entail further investigation into practical stopping rules for the algorithm as discussed briefly at the end of Section 3. We also need to experiment with the choice of the DIRECT threshold parameter ε appearing in (3.2). In the above examples we have used the typical value 5×10^{-4}, which is smaller than the value recommended in [11] but which seems necessary for the route-finding problem in order to force the method to approach the global solution. One other parameter which has an important effect on runtimes for solving the aircraft routing problem is the value σ_{max} used in the exploration of each leg to determine threat violations. For the above examples we have used $\sigma_{max} = 4$ km, which might be considered rather coarse but which nevertheless seems to give results in good agreement with routes obtained with a smaller step size. In general, it might be advisable to decrease σ_{max} on each cycle of DIRECT as the stage lengths tend to become shorter.

As a final and more general observation, we note that DIRECT has some similarities with interval arithmetic methods of global optimization (see the recent survey by Wolfe [15], for example) in that it selects potentially optimal hyperboxes on the basis of estimated bounds on the objective function. In principle, evaluating an interval extension of f on a hyperbox will yield valid bounds on f rather than speculative ones involving unknown Lipschitz constants. However, this apparent advantage must be set against *(a)* the relatively high cost of interval computations compared to standard real arithmetic and *(b)* the fact that interval arithmetic bounds, although valid, are often pessimistic. Moreover, it is not clear that interval arithmetic arithmetic is applicable to the calculation of l_{ji} by sampling as discussed in section 2. However, for objective functions whose interval extension is available, an interesting topic for further study, therefore, would be a numerical comparison of DIRECT with some of the interval techniques outlined in [15].

References

[1] K.S. Al-Sultan and M.A. Al-Fawzan, A Tabu Search Hooke and Jeeves Algorithm for Unconstrained Optimization, European Journal of OR, 103, 198-208, 1997

[2] C.A. Baker, L.T. Watson, B. Grossman, R.T.Hafka and W.H. Mason, Parallel Global Aircraft Configuration Design Space Exploration, Proceedings of the High Performance Computing Symposium 2000 (ed A.Tentner), 101-106, Society for Computer Simulation International, San Diego, 2000.

[3] C.A. Baker, L.T. Watson, S.E. Cox, B. Grossman, R.T.Hafka and W.H. Mason, Study of a Global Design Space Exploration Method for Aerospace Vehicles, 5th NASA High Performance Computing and Communicationa Computational Aerosciences Workshop, NASA, Mountain View, 2000.

[4] C.A. Baker, L.T. Watson, B. Grossman, W.H. Mason and R.T. Hafka, Parallel Global Aircraft Configuration Design Space Exploration, Int. J. Comput. Res. (to appear).

[5] R. Chelouah and P. Siarry, Tabu Search Applied to Global Optimization, European Journal of OR, 123, 256-270, 2000

[6] S.E. Cox, R.T. Hafka, C.A. Baker, B. Grossman, W.H. Mason and L.T. Watson, Global Optimization of a Highh Speed Civil Transport Configuration, 3rd World Congress of Structural and Multidisciplinary Optimization, Buffalo, 1999.

[7] S.E. Cox, R.T. Hafka, C.A. Baker, B. Grossman, W.H. Mason and L.T.Watson, Global Multi-disciplinary Optimization of a High Speed Civil Transport, J. Global Optim (to appear)

[8] J.M. Gablonsky, An Implementation of the DIRECT Algorithm, Tech. Report CRSC-TR98-29, Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC, 1998.

[9] C. Hewitt and S.A. Broatch, A Tactical Navigation and Routeing System for Low-level Flight, Technical Report, GEC-Marconi Avionics, Rochester, Kent, U.K. (AGARD, Italy, 1992)

[10] C. Hewitt and P. Martin, Advanced Mission Management, Technical Report, GEC-Marconi Avionics, Rochester, Kent, U.K. (IEE - FITEC, 1998)

[11] D.R.Jones, C.D. Perttunen and B.E.Stuckman, Lipschitzian Optimization without the Lipschitz Constant, Jounal of Optimization Theory and Applications, 79, 157-181, 1993

[12] L.T. Watson and C.A. Baker, A Fully Distributed Parallel Global Search Algorithm, 10th SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 2001.

[13] L.T. Watson and C.A. Baker, A Fully Distributed Parallel Global Search Algorithm, Engrg. Comput., (to appear)

[14] S.P. Wilson, S.C. Parkhurst and M.C. Bartholomew-Biggs, The Aircraft Routing Problem, Technical Report 331, Numerical Optimisation Centre, University of Hertfordshire, December 2000

[15] M.A. Wolfe, Interval Mathematics, Algebraic Equations and Optimization, J. Computational and Applied Mathematics, 124, 263-280, 2000