A GLOBAL OPTIMIZATION APPROACH TO SOLVE MULTI-AIRCRAFT ROUTING PROBLEMS

S.P. Wilson, M.C. Bartholomew-Biggs and S.C. Parkhurst

Numerical Optimisation Centre, University of Hertfordshire, United Kingdom.

(This work was supported by BAE SYSTEMS, Rochester, England)

Keywords: Global optimization; Direct-search methods; Aircraft routing; Route Planning; DIRECT algorithm.

Abstract

This paper describes the formulation and solution of a multi-aircraft routing problem which is posed as a global optimization calculation. The paper extends previous work (involving a single aircraft using two dimensions) which established that the algorithm DIRECT is a suitable solution technique. The present work considers a number of ways of dealing with multiple routes using different problem decompositions. A further enhancement is the introduction of altitude to the problems so that full three-dimensional routes can be produced. Illustrative numerical results are presented involving up to three aircraft and including examples which feature routes over real-life terrain data.

1. Introduction

The aircraft routing problem involves finding paths (in two or three dimensions) from a specified start point to a specified destination (possibly also passing through some rendezvous points on the way). Routes are composed of straight line segments joining intermediate "waypoints" and the problem is to position these waypoints so that the resulting path avoids certain "hard" constraints (such as geographical features and "no-fly zones") and has low exposure to "soft" constraints (such as risks from military missile threats or low fuel reserves). The *optimality* of a route is based on minimizing a "cost" composed of a weighted sum of penalties for constraint violations together with similar penalties for lateness at rendezvous points, flying excessive distances, using extreme manoeuvres et cetera. More discussion about operational aspects of aircraft routing can be found in (Hewitt & Broatch, 1992) and (Hewitt & Martin, 1998) which also describe a heuristic routing algorithm. Other papers which discuss different versions of the aircraft routing problem include (Chen et al., 1995; Zabarankin et al., 2006; Karczewski, 2007 and Carlyle et al., 2007).

This paper is an extension of earlier work (Bartholomew-Biggs, Parkhurst & Wilson, 2003, both papers) in which the aircraft routing problem was tackled using a number of global optimization direct search techniques applied to a basic two-dimensional problem which we shall refer to as the Simplified Route Model (SRM). It was found that the deterministic DIRECT algorithm (Jones, Perttunen & Stuckman, 1993) appeared better suited to this

problem than the random-sampling and tabu-search techniques of TSHJ (Al-Sultan & Al-Fawzan, 1997) and ECTS (Chelouah & Siarry, 2000).

Our previous work has been concerned with finding routes for a single aircraft. This paper extends the SRM cost function both to handle multiple routes and also to introduce altitude to the problem so that full three-dimensional routes can be produced. We present results using both the SRM and also a Realistic Route Model (RRM) (Wilson, 2003), involving actual terrain data.

The SRM test results given in this paper were obtained from a Pentium 3 (500Mhz) PC running on a Linux platform. Hardware details for the RRM model cannot be disclosed. All of the software was implemented in C/C++.

2. The Simplified Route Model (SRM) route cost function

We first summarise the main features of the Simplifed Route Model; more details can be found in (Wilson, 2003). In the two-dimensional case we can attempt to find the ground plan of a route, avoiding a number of obstacles that we shall henceforth call "threats". A route will be defined by its (given) start and end points and by a number of intermediate waypoints. The co-ordinates of these waypoints will be optimization variables; and we assume that the flight path follows straight lines between them. We shall now describe a Simplified Route Model (SRM) route cost calculation.

Suppose that the distance flown is to be as short as possible, subject to suitable avoidance of the threats. Then, for any choice of waypoints, we first calculate the Euclidean length of the corresponding route, denoted by *L*, say. We can write $L = \sum l_j$ where l_j is the length of the "leg" between the *j*-th and (*j*+1)-th waypoint. We then determine how much of the route passes through threats. If the route passes through the i-th threat for a distance L_i then the "cost" of the route C can be expressed as

$$C = L + \sum_i \rho_i L_i^p \qquad (1)$$

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 waypoint co-ordinates so as to minimize C. The balance between flight path length and threat penetration will depend upon the choice of the parameters ρ_i. If the i-th threat is a physical obstacle then ρ_i must be large to ensure that the solution does not attempt to pass through it; but if threat i represents some risky but not impossible region then a more moderate value of ρ_i may be appropriate because it will allow a shorter route which makes an acceptably brief incursions into some danger area. Hence, in (1), the length of leg j that lies inside threat *i* is defined as $L_i = \sum l_{ji}$.

In the SRM examples used in this paper we shall use circular threats; but in general we may need to deal with no-fly zones and geographical obstacles with 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 a 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 database which provides terrain height for given latitude and longitude will enable us to determine whether some constant altitude flight-path is feasible or not). Then, using a sampling and linear interpolation process described in (Bartholomew-Biggs et al., 2003) and (Wilson, 2003), we can calculate a value for l_{ji}.

In practice we wish to avoid routes which involve sharp turns and short distances between waypoints. Therefore we can add further penalty terms to the cost function to reflect these issues. Let ϕ_j denote the angle between legs j and $j+1$; if $\phi_{\max}$ is the limiting turn angle then we define

$$\psi_j = \begin{cases} 0 \text{ if } \phi_j \le \phi_{\max} \\ \phi_j - \phi_{\max} \text{ otherwise} \end{cases}$$

If $l_{\min}$ denotes the least acceptable leg-length then we can also define

$$\bar{l}_j = \begin{cases} 0 \text{ if } l_j \ge l_{\min} \\ l_{\min} - l_j \text{ otherwise} \end{cases}$$

Taking the above ideas into account, the route cost function that we use for a problem with *n* stages and *m* threats is slightly different from (1) in that it individually penalises stage lengths lying inside threats. The function is

$$C = \sum_{j=1}^{n+1} (l_j + \mu \bar{l}_j^2 + \sum_{i=1}^{m} \rho_i l_{ji}^3) + \sum_{j=1}^{n} \nu \psi_j^2 \qquad (2)$$

where μ and ν are penalty parameters. We have chosen the threat penalty exponent as $p = 3$ because it is shown in Wilson, 2003 that, provided $p \ge 3$, C retains continuous first derivatives as changes in waypoint position cause a leg to move from being wholly outside a threat to being partly inside it . The usual quadratic penalty for the other constraints ensures that C is differentiable at boundaries of the feasible region.

3. The Realistic Route Model (RRM) route cost function

The Realistic Route Model (RRM) cost function uses real geographical data to define physical obstacles and also involves more types of constraint than the SRM. These include no-go areas, weather, exposure to military threats and fuel limitations. Evaluations of a route cost depends upon reference to a database of terrain information in order to evaluate penalties associated with flying too close to obstacles or with over-exposure to threats such as radar detection. The precise details of this route cost function (the relative weights attached to different constraints and the ways in which these constraints are measured) may not be disclosed. However, they can be assumed to reflect considerable operational experience

within the aerospace industry. We shall be using this route cost evaluation as a “black box” in examples discussed in later sections.

4. Global optimization approach and the DIRECT algorithm

It is a feature of both the SRM and RRM route cost functions that they often have several local minima. This is easily understood if we consider two possible routes that pass either side of some obstacle: there may be a locally "best" route for each of the alternative branches but these will be separated by routes which have a high value of the cost function. In practice we want to find the best of all the locally optimal routes, i.e., the global optimum.

We now give a brief description of the global optimization algorithm DIRECT (Jones et al., 1993). This is a deterministic rather then random approach which works by repeatedly dividing the region of search into smaller and smaller hyperboxes.

When applied to a function *f(x)*, DIRECT begins with a given "hyperbox" defined by upper and lower bounds on the optimization variables. At the beginning of the first iteration we have the value of the objective function at a centre point, i.e., $f_0 = f(c_0)$. The initial region is then systematically split into smaller hyperboxes on subsequent iterations. For each hyperbox j ($j = 1,\ldots,J$) we have a centre point c_j (where the function value is f_j). Hyperboxes are grouped according to a size parameter δ_j, which is the distance from its centre to any corner.

At each iteration of DIRECT, some of the current hyperboxes $j = 1,\ldots,J$ are selected for further subdivision. The aim is to explore the whole region efficiently by only computing extra function values in regions which can be termed "potentially optimal". Potentially optimal hyperboxes are chosen via a procedure given below. We shall suppose that among the J hyperboxes there are only $K_J \leq J$ different size values. We note first of all, however, that we need only examine K_J of the hyperboxes – i.e. for each of the different δ_j-sized candidates, we need only consider the one whose centre has the least function value. Subdivision of a hyperbox involves trisecting it along the longest edge. (If several edges have the same "longest" length, then the trisection process is repeated for each of them

We now explain the process of selecting hyperboxes which are worth further exploration. If Ω is a Lipschitz constant for *f* then a lower bound upon *f* inside the hyperbox j is given by $f_{\_j} = f_j - \Omega\delta_j$. Hence, the most promising box would be the one for which $f_{\_j}$ is smallest. Since a valid Lipschitz constant will not usually be known, DIRECT is based on considering whether there exists any Lipschitz constant such that box j *could* contain a lower function value than any other box. Thus box j is potentially better than box k if there exists a Lipschitz constant Ω such that

$$f_j - \Omega\delta_j \leq f_k - \Omega\delta_k .$$

(No such Ω exists if $\delta_j = \delta_k$ and $f_j \geq f_k$ and hence, as already mentioned, we need only test the potential optimality of boxes having the smallest *f* value for any size parameter δ.)

If $\delta_j > \delta_k$, then box j can only be potentially optimal for

$$\Omega \geq \Omega_{\min_k} = \frac{f_j - f_k}{\delta_j - \delta_k}$$

while if $\delta_j < \delta_k$ then box j can only be potentially optimal for

$$\Omega \leq \Omega_{\max_k} = \frac{f_j - f_k}{\delta_j - \delta_k}.$$

Thus, for a box to be potentially optimal, the following condition must be met:

$$\Omega_{\min} \leq \Omega \leq \Omega_{\max} \text{where } \Omega_{\min} = \max(\Omega_{\min_k}) \text{ and } \Omega_{\max} = \min(\Omega_{\max_k}).$$

Hence, there can only be a suitable Ω if $\Omega_{\max} > \Omega_{\min}$ and $\Omega_{\max} > 0$. If these conditions do not hold, then box j cannot be potentially optimal.

To reduce the number of boxes to be subdivided, a further filter is applied. We only treat box j as potentially optimal if

$$f_j - \Omega_{\max}\delta_j < f_{\min} - \varepsilon|f_{\min}|.$$

If this inequality fails then box j is not judged to be worth further subdivision. In the original version of DIRECT, ε is the only user-choice to be made. In the examples which follow, we have typically used ε=0.01 as suggested by the authors (Jones et al., 1993). No automatic convergence test is proposed by the authors; instead it is suggested that the algorithm should be simply run for a fixed number of iterations.

Refinements to DIRECT

In applying DIRECT to routing problems we make use of various improvements to the algorithm outlined above. These have been proposed by (Bartholomew-Biggs et al., 2003; He et al., 2002; Jones, 2001; Watson & Baker, 2001).

Stopping criteria

One form of stopping rule is based on terminating the algorithm when the objective function does not decrease sufficiently over a specified number of iterations. However, employing such a criterion carries a risk of premature termination. It can be quite common for DIRECT not to decrease the objective function for a significant number of steps while it explores a region round a local optimum. It may take many iterations before potentially optimal hyperboxes are identified in a new area (which could contain the global solution).

Another stopping criterion proposed by He et al. (2002) is to stop DIRECT when boxes become sufficiently small. This idea can be used to save wasteful function evaluations by

preventing subdivisions from occurring when box j has $\delta_j < \delta_{\min}$ where $\delta_{\min}$ is a user-defined parameter.

Single box subdivisions

As proposed by Jones (2001), performing only one subdivision of each potentially optimal hyperbox can improve the economy of the algorithm. In high dimensional problems where the original hyperbox has several edges the same length, the original version of DIRECT may subdivide potentially optimal hyperboxes many times on each iteration which increases the number of function evaluations. Because not all potentially optimal hyperboxes will actually be worth exploring, this may produce a lot of redundant evaluations. Performing single hyperbox subdivisions will mean that regions that are really promising will be eventually identified; in the meantime, the number of new boxes and function evaluations will not grow quite so rapidly. This will give a saving in both memory and computational workload.

Aggressive searches

Another possible change to DIRECT (Watson et al, 2001) is to perform periodic "aggressive searches" in which *all* the candidate hyperboxes are subdivided without using the filter on potential optimality. The aim is obviously to cause exploration of the neighbourhood of the global optimum to start after fewer iterations. Any benefits will, however, come at the cost of creating a bigger set of new boxes at each aggressive iteration, leading to an increase in the computational workload and storage requirements.

Use of restarts

In the computational tests described in Bartholomew-Biggs et al., 2003; Wilson, 2003 it has been noted that it is often beneficial to use DIRECT in "restart" mode. That is, instead of performing *m* iterations from one initial centre point and within one initial hyperbox, we can stop after, say, *m/*2 iterations and begin a new DIRECT exploration in a hyperbox centred on the best point found so far. It is often the case that performing *2*m/2* (or perhaps *4*m/4*) iterations in this way will yield a better point (and at lower computational cost) than what would be obtained by *m* straight iterations of DIRECT. Possible reasons for this are discussed in Wilson, 2003.

DIRECT versions

In some of the examples below, we quote results from DIRECT-1, which is an implementation of DIRECT including the minimum box size stopping criteria (with $\delta_{\min}$ = 0.001) and also using aggressive searches. An aggressive search is applied if no significant decrease in $f_{\min}$ occurs after $100n$ function evaluations. At most, two aggressive searches are allowed and thereafter the algorithm stops if no reduction is observed in $f_{\min}$ after $100n$ function evaluations.

A second implementation, DIRECT-2, is also included in some of our tests. It uses all the features of DIRECT-1 but performs a single subdivision of each potentially optimal box.

DIRECT-2 is therefore less expensive per iteration than DIRECT-1 (and because of this we will allow it to perform more iterations).

5. Using DIRECT for multi-aircraft out-and-home problems

Additions to the SRM route model

The examples in this section are all based on the two-dimensional SRM cost function. In order to deal with multiple aircraft we form a cost function for each route and then consider the minimization of the sum of these functions. However, we must also add a penalty term that ensures that the aircraft do not come too close together. To do this, we need to consider the speed of each aircraft (which is assumed to remain constant throughout a mission). It is then possible to calculate the position of the aircraft along their respective paths after *t* seconds and the distance between them can be checked to ensure that it exceeds the minimum allowable value.

In the examples presented in this paper, we assume that the aircraft are together at time $t = 0$ and we use an "exclusion zone" near the start of the mission within which there is no penalty for aircraft being too close together. In practice the aircraft could be started at different times (probably with the faster aircraft leaving later). For the purposes of the investigation in this paper we shall not deal with such operational decisions.

Approaches to solving multi-route problems

In our experiments – described fully in Wilson, 2003 – we have chiefly been interested in planning missions involving both outward and return phases. These have generally proved to be more challenging than "one-way" routing; and three different approaches have been tried.

Approach 1 solves for all routes simultaneously and regards the waypoints for all *n* routes as variables in one optimization problem. This approach seems adequate for problems with small numbers of waypoints (e.g. those involving "one-way" rather than "round trip" routes). However, as the number of waypoints increased many DIRECT iterations were required before acceptable routes were found. This in turn led to very large computational costs.

Approach 2 involves dealing with outbound and return (multi-aircraft) routes separately. This approach has not worked well because the separate computation of out-and-home routes does not take account of the "whole picture". When DIRECT is optimizing the outbound routes it will aim to find the best possible routes between the start and target. These optimum outbound routes, however, may not be well positioned to permit good return paths.

Approach 3 (the most successful one) is to solve the whole out-and-return problem separately for each aircraft. This was motivated by our difficulties with approach one. Considering one route at a time will make the problem size smaller than if we were optimizing several routes at the same time. Our experiments (Wilson, 2003) showed that approach three produces much better routes than the others and the computational cost is smaller.

In all the following examples, therefore, we use the approach of solving whole out-and-return problems separately. The route separation constraints are implemented as follows. The first route is computed by DIRECT with the separation constraints disabled. Once the first route

has been generated, the waypoint positions are stored. The second route is then computed by DIRECT (using the same starting guess as the preceding route) but this time with the separation constraints enabled. Every time a route cost is requested, the stored waypoints from the previously computed route (or routes) are used to determine any separation penalties.

Solving multi-route out-and-home problems

In this section we give results obtained by DIRECT on various multi-aircraft SRM problems. The penalty factors appearing in a cost function like (2) for all the problems in this section are: Threat penalty:10; Turn angle penalty: 5; Leg length penalty:1; Route separation penalty:10. (The turn angle penalty factor is relatively high to prevent unacceptable turn angles at the target point during transition from outbound to homebound route.) The separation penalty exclusion zone has radius 1km.

All problems involve aircraft flying at equal speeds since this is more challenging for the separation constraint. The initial guess used for all the problems places waypoints at equal intervals along a stright line from start to target. The initial box sizes for each waypoint coordinate was ± 10km which allows DIRECT ample freedom to move the waypoints into optimal positions. For all the problems, we have run DIRECT in restart mode (see section 4).

Test 5.1 – Solving a five-threat problem involving two aircraft

We start by considering a five-threat problem in which eight waypoints are used for each route. We allow 4*128 DIRECT iterations to be performed for each route computation. The routes found are shown in the Figure 5.1 below.

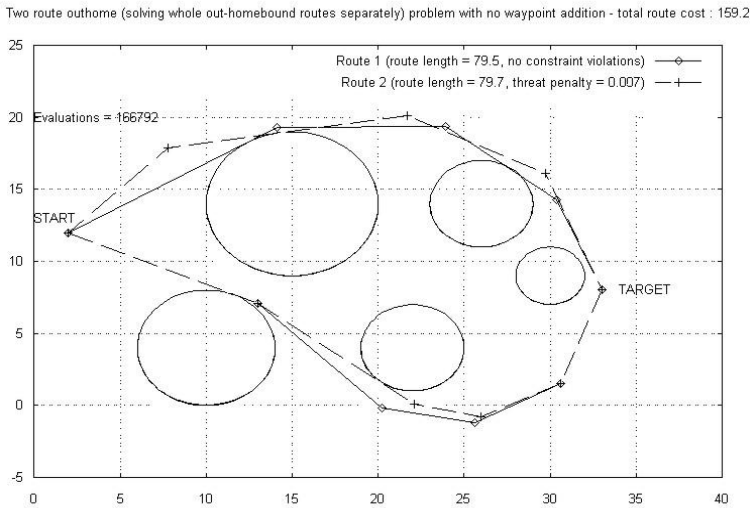

Figure 5.1

Two very good routes have been found. Both routes initially diverge because of separation constraints and converge towards the end of the return route because, by then, one aircraft has fallen behind the other. The total number of evaluations used was 166792 and each restart took eight minutes. This compares very well with the simultaneous route-finding approach 1 which did nearly twice as much work and gave routes that were significantly poorer than those in Figure 5.1.

Test 5.2 – Solving a ten-threat problem involving two aircraft

We next consider a ten-threat problem using twelve waypoints for each route. We again allow 4*128 DIRECT iterations for each route computation. Figure 5.2 shows that both the routes found avoid all the threats. As with the previous test, both routes have remained fairly close to each other throughout the entire mission.

A total of 288434 evaluations were used to obtain the solution shown in Figure 5.2 and the time taken for DIRECT to run each restart was approximately twenty minutes. This is once again a significant improvement over approach 1 which produced worse routes and took 543026 evaluations.

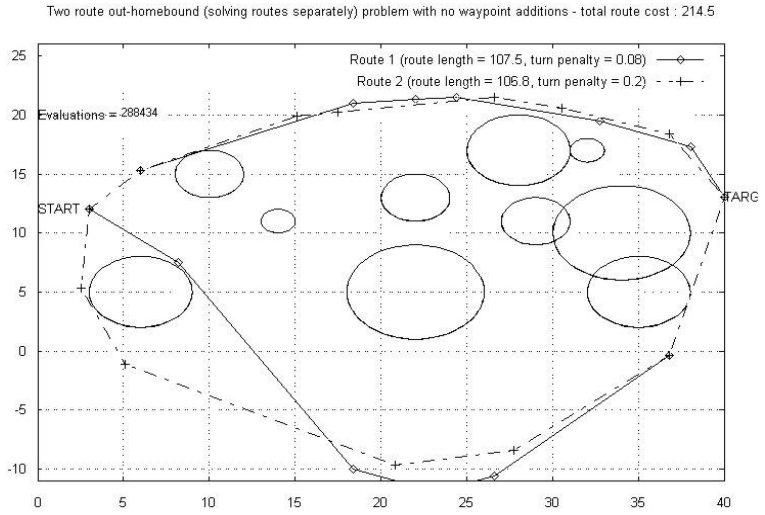

Figure 5.2

In Tests 5.1 and 5.2 the aircraft have remained fairly close throughout the mission. Because the same starting guesses are used for each route, the initial subdivisions of the hyper-boxes surrounding the waypoints are identical and so the same areas are searched for each route calculation. This tends to mean that the routes will only move apart if the minimum separation constraint forces them to do so.

In other tests – not detailed here - we also introduced a *maximum* separation constraint to the problem to prevent routes from separating too much. Keeping aircraft reasonably close to one another can be quite an important mission requirement since it allows mutual protection from attacks by enemy fighters and provides more personnel to monitor any ground or air threats. We had some partial success with this maximum separation constraint. In particular, we found that where routes would normally separate and fly around different sides of a threat, the constraint forced the routes together so that they passed on the same side. However, little difference was made to routes that took similar directions around threats. In such cases, the maximum separation constraint sometimes forced the routes to be slightly shorter. More detailed information on the maximum separation constraints can be found in (Wilson, 2003). It is worth commenting here, however, that careless imposition of both maximum and minimum separation constraints can result in a problem having no feasible solution!

Using automatic waypoint additions

In the previous examples we assumed that we knew in advance how many waypoints would be "enough" to enable a route to negotiate the threats and obstacles. We now describe an automatic strategy for adding extra waypoints during a solution. This involves examining the routes obtained by DIRECT at the end of a restart cycle and adding an extra waypoint at the midpoint of any leg that passes through a threat. The enclosing box of this new waypoint has the same size as that for the old waypoints. (New waypoints are only added to route legs whose length is greater than $2\,l_{\min}$.)

Extensive experimentation with the use of automatic waypoint additions has shown that good routes can be produced with low computational cost when we start off with a small number of waypoints and only allow additions to occur on alternate restarts of DIRECT. This strategy enables DIRECT to conduct a more thorough search with a current set of waypoint variables before any new ones are added. We now give an example of the waypoint additions strategy.

Test 5.3 – Solving a five-threat problem using waypoint additions

This is the same problem as Test 5.1, but the DIRECT solution is now started with just three waypoints placed at equal intervals along the straight line between the start and destination. The routes found by DIRECT are shown in Figure 5.3; Table 5.1 compares these results with those from Test 5.1.

Compared with Test 5.1, Test 5.3 needed two more restarts of DIRECT, along with an extra call to the automatic waypoint addition procedure, before route 2 became feasible. An improvement is seen in the length of route 1 when compared to Test 5.1 but this has resulted in slight violations to the constraints. Route 2 however, is not as good as route 2 in Test 5.1. It should be understood, however, that such differences between individual routes may be rather accidental, since DIRECT is seeking to minimize the *total* route cost: and Table 5.1 shows that the sum of the route lengths is quite similar for Test 5.1 and Test 5.3. Nevertheless, we can still regard the waypoint-addition approach as successful because, by starting off with fewer waypoints, we have solved Test 5.3 more cheaply than Test 5.1.

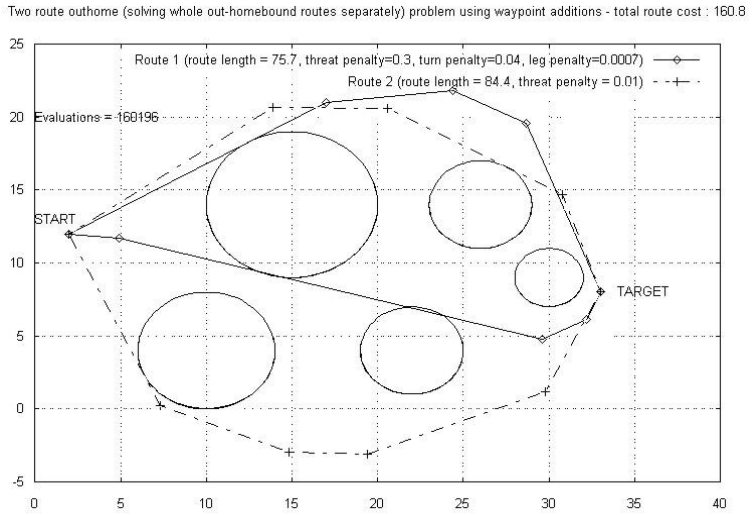

Figure 5.3

Test	Route number	length	Threat	Turn	leg	Total evaluations
5.1 Using 8 waypoints with no waypoint additions	1	79.5	0	0	0	166792
	2	79.7	0.007	0	0	
5.3 Starting with 3 waypoints and using waypoint additions	1	75.7(4)	0.3	0.04	0	160196
	2	84.8(5)	0.01	0	0	

*The value in parenthesis shows the number of added waypoints

Table 5.1 – Comparing results of tests 5.1 and 5.3

Test 5.4 – Solving a ten-threat problem using waypoint additions

We next consider the ten-ten threat problem from Test 5.2. Starting with four waypoints (equally spaced along the straight line between the start and destination) and allowing waypoint additions after every other restart of DIRECT we obtained the routes shown in Figure 5.4. A comparison between the solutions to Tests 5.2 and 5.4 appears as Table 5.2.

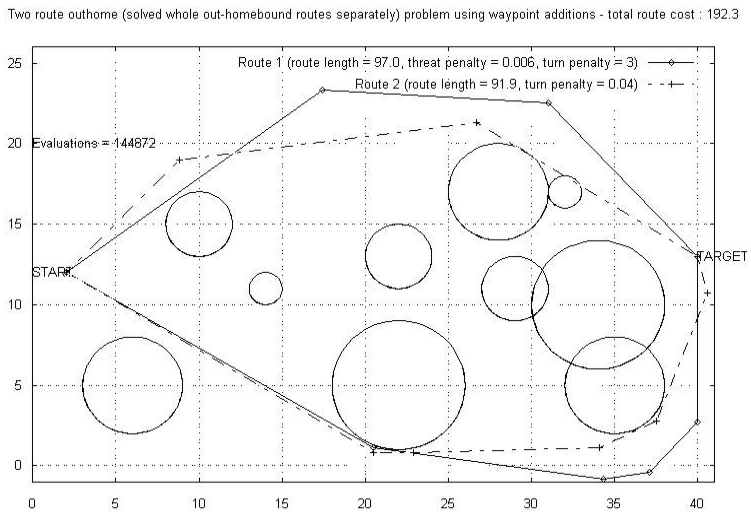

Figure 5.4

Test	Route number	length	Threat	Turn	leg	Total evaluations
5.2 Using 12 waypoints with no waypoint additions	1	107.5	0	0.08	0	288434
	2	106.8	0	0.2	0	
5.4 Starting with 4 waypoints and using waypoint additions	1	97.0(3)	0.006	3	0	144872
	2	91.9(4)	0	0.04	0	

*The value in parenthesis shows the number of added waypoints

Table 5.2 – Comparing results of Tests 5.2 and 5.4

In this case, use of the waypoint addition strategy has produced much better routes than those obtained with a fixed number of waypoints. Moreover, Test 5.4 incurs far less computing cost than Test 5.2 both because it uses half the number of function evaluations and also because it involves fewer waypoints and so each DIRECT iteration will be less expensive. The only drawback in the solution to test 5.4 is that Route 1 has incurred some minor turn penalties.

6. Multi-aircraft problems involving the RRM cost function

In this section we present some multi-aircraft routes obtained when DIRECT is applied to the RRM cost function using the one-route-at-a-time strategy. In this section we make use of the modified versions of DIRECT, introduced at the end of section 4, and known as DIRECT-1 and DIRECT-2.

We consider the three scenarios all using the same terrain, threats and no-fly zones but characterised by different start and finish points. For each scenario, we compute outbound only routes for three aircraft flying at equal speeds. For all tests we set the minimum separation to 200 metres (allowing a separation penalty exclusion zone of 1km near the start). For each scenario, initial guessed waypoints equi-spaced along the straight line joining the start and target.

Scenario 1

We start by running the three versions of DIRECT using Scenario 1. For this problem, a box size equivalent to about 63km north-south and 40km east-west is used. We allow DIRECT and DIRECT-1 to run for 4*32 iterations per route and DIRECT-2 for 4*128 iterations per route. Although 4*32 iterations seems a small number, our experience indicates that this is adequate to find good routes on this scenario. Results are summarised in Table 6.1.

	DIRECT		DIRECT-1		DIRECT-2	
Route	Route cost	Evals	Route cost	Evals	Route cost	Evals
1	445.6	57952	383.6	64634	457.3	31130
2	387.7		453.0		460.5	
3	374.2		392.8		559.9	

Table 6.1 – Performance of DIRECT for RRM cost function on Scenario 1

Of the three variants, the basic version of DIRECT has performed best although the total route cost is only about 0.2% less than that found by DIRECT-1. The routes produced (see Figure 6.1) are obtained at smaller computational cost. DIRECT-2 was the cheapest variant and found routes which – although visually similar to those found by DIRECT and DIRECT-1 – had route costs about 2% higher. We can see from Figure 6.1 that all three routes follow very similar paths to the target and appear to make sense in terms of avoidance of threats

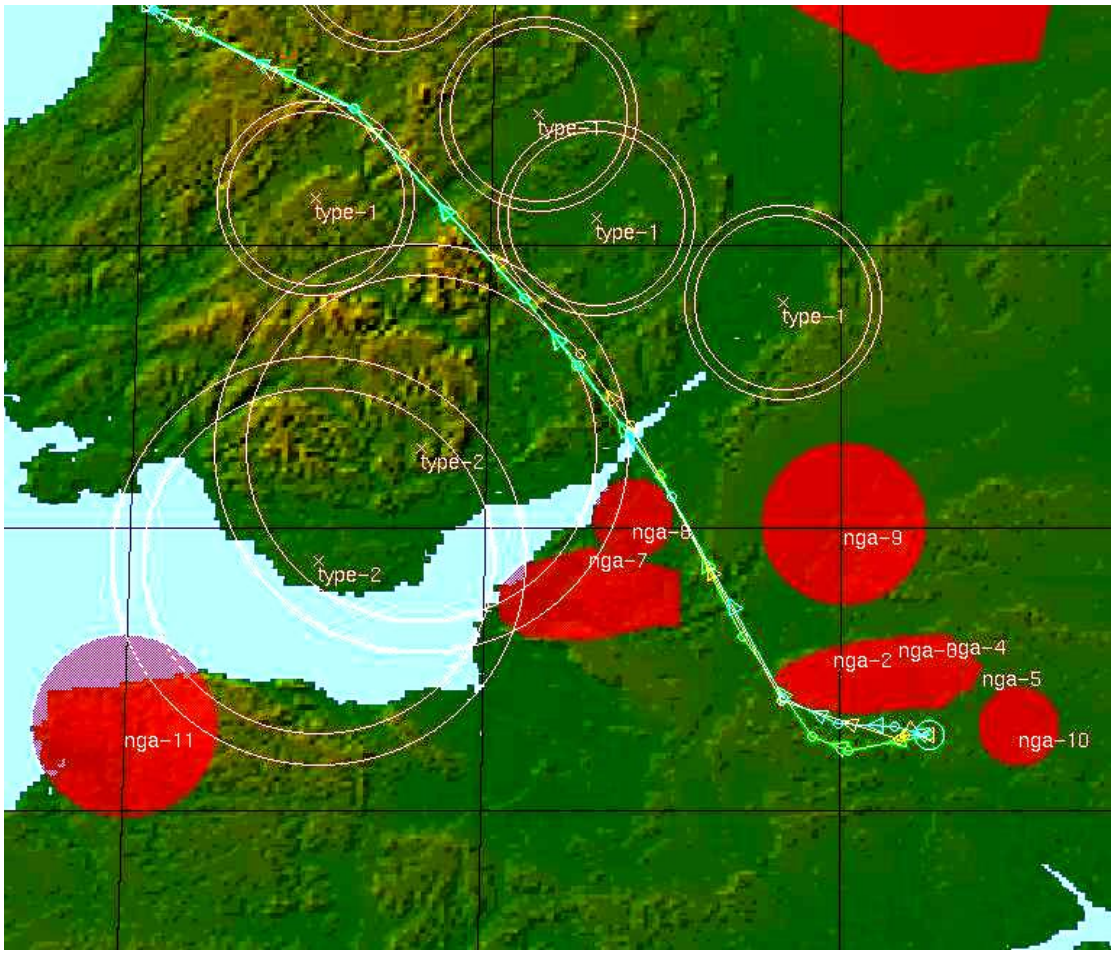

Figure 6.1 – Routes produced by DIRECT on Scenario 1

Scenario 2

For this problem we use a box approximately 120km north-south and 80km east-west and allow DIRECT and DIRECT-1 to run for 8*32 iterations and DIRECT-2 to run for 8*128 iterations. The routes found by each variant are shown in Table 6.2.

	DIRECT		DIRECT-1		DIRECT-2	
Route	Route cost	Evals	Route cost	Evals	Route cost	Evals
1	3234.3	49446	1245.6	54306	5333.3	41260
2	1151.5		1234.2		4213.4	
3	1128.3		1163.4		4626.7	

Table 6.2 – Performance of DIRECT for RRM cost function on Scenario 2

For this scenario, DIRECT-1 has produced much better solutions (see Figure 6.2). DIRECT found good solutions for routes 2 and 3 but did not do so well for route 1. DIRECT-2 has performed the worst by finding routes that are not at all competitive with the other two versions.

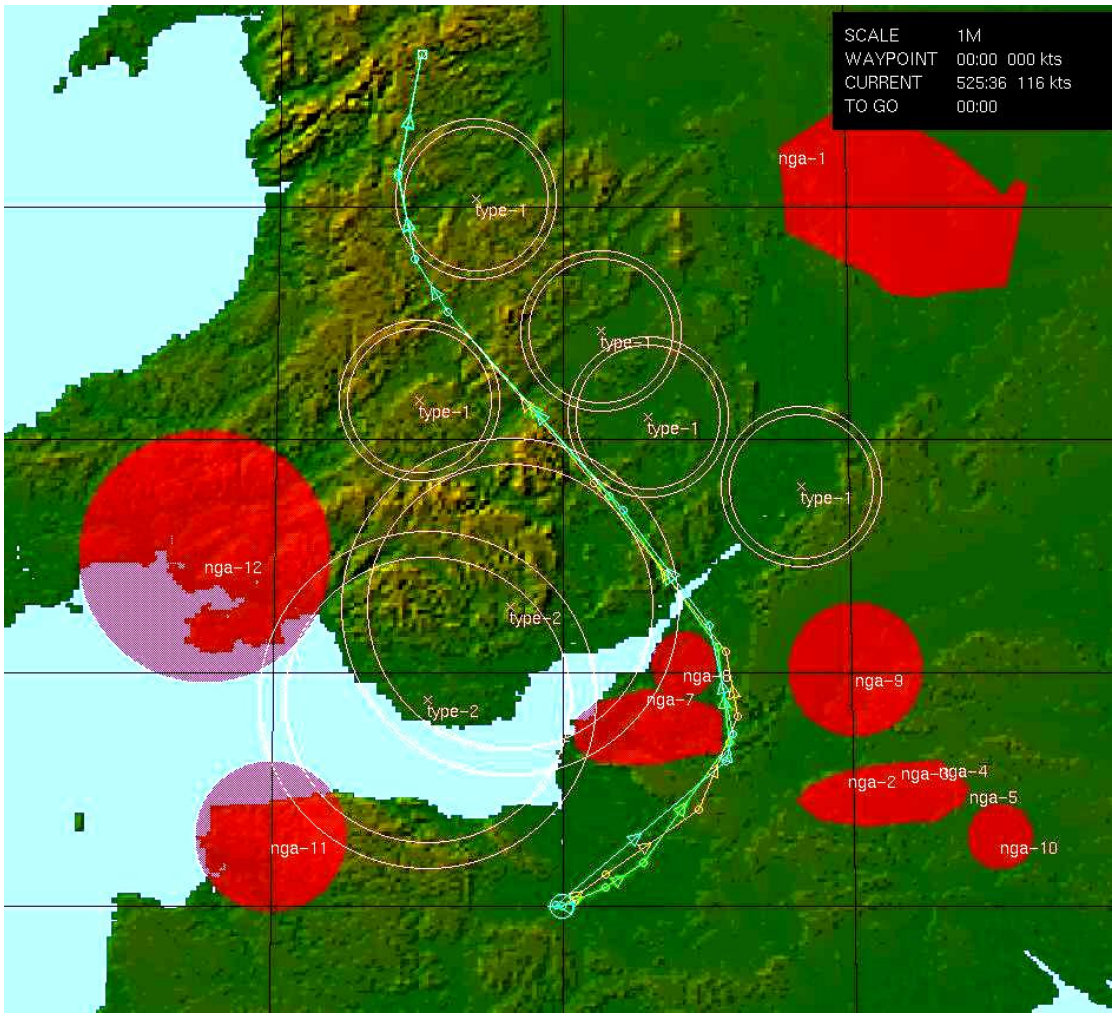

Figure 6.2 – Routes produced by DIRECT-1 on Scenario 2

As with the previous case, all three routes appear visually sensible and follow very similar paths to the target. The initial separation of the routes will mean that they can follow in line at the later stages.

Scenario 3

For this problem we use the same box–size as scenario 1 and allow DIRECT and DIRECT-1 to perform 4*64 iterations per route and DIRECT-2, 4*128 iterations per route. The solutions found are given in Table 6.3. On this scenario DIRECT has performed best (see Figure 6.3). DIRECT-1 produces similar routes, with the exception of route 1, which has a much higher cost. DIRECT-2 has again performed worst but used far fewer evaluations than the other versions. A further test was performed on this scenario in which DIRECT-2 was run for 4*256 iterations. The solutions produced were much improved and competitive with those found by DIRECT and the number of evaluations used was 105896.

	DIRECT		DIRECT-1		DIRECT-2	
Route	Route cost	Evals	Route cost	Evals	Route cost	Evals
1	1609.8	115486	2756.6	134850	2586.2	55308
2	2551.8		2273.4		2574.6	
3	2522.3		2537.3		3856.2	

Table 6.3 – Performance of DIRECT for RRM cost function on Scenario 3

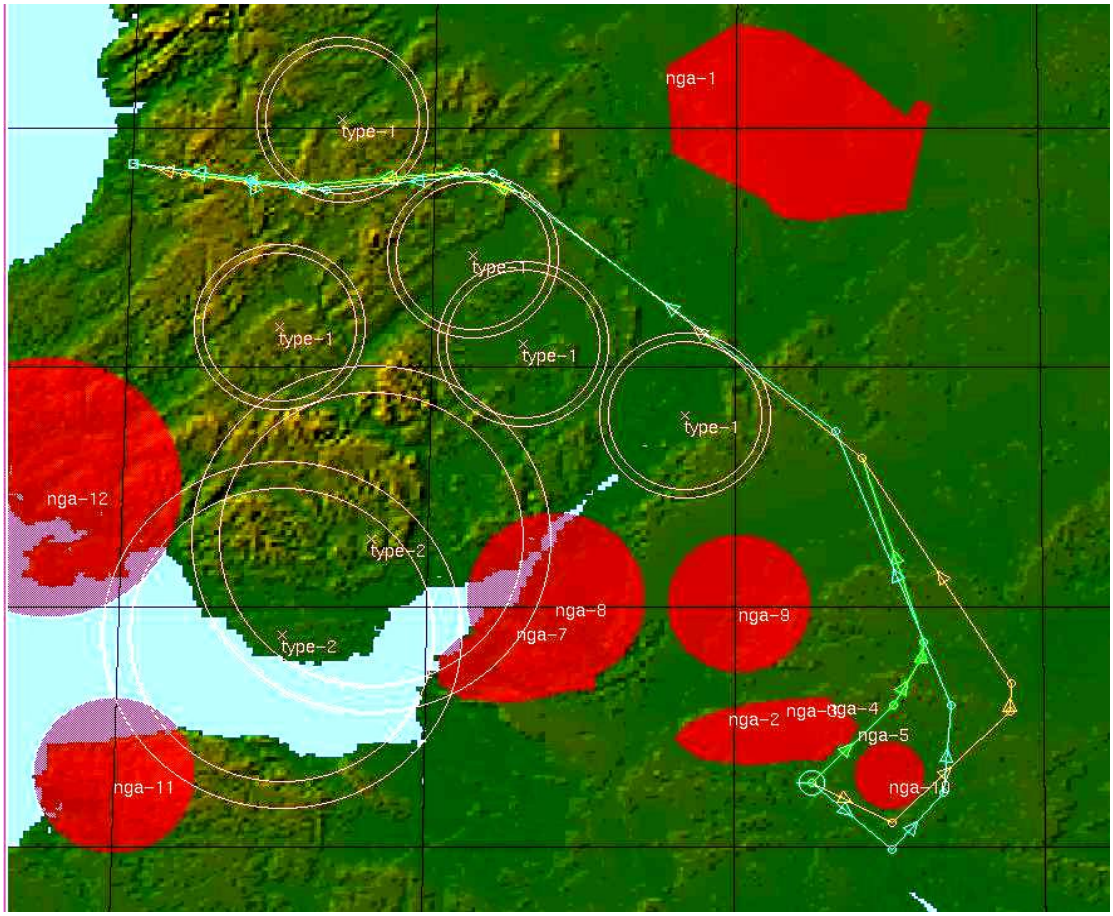

Figure 6.3 – Routes produced by DIRECT on Scenario 3

DIRECT appears to have found routes that are only locally optimal since the global optimum lies in the middle valley between the threats (the position of the global optimum had been previously determined through numerical experiments). The initial legs of route 1 pass between the two no-go areas whilst the initial legs of routes 2 and 3 are forced around the outside of the small no-go area (at the bottom of the figure) to satisfy the separation constraint.

7. Extending multi-aircraft routing to three-dimensions

In this section, we extend the routing problem to three space dimensions and investigate the performance of DIRECT (and its variants) on SRM problems involving altitude. Routes are now composed of straight line segments between waypoints expressed as (*x,y,z*) coordinates.

Adding an altitude dimension to threats

To account for the extra dimension, some threats are represented by ellipsoids. This allows for modelling geographical obstacles like hills or mountains of varying height. Threats such as missile sites can be made spherical and no-fly zones could be modelled using infinite cylinders.

Threats are now defined by centres $c_i = (c_{ix}, c_{iy}, c_{iz})$ and radii $r_i = (r_{ix,} r_{iy}, r_{iz})$. Thus, a point (x, y, z) on the path is inside a threat if

$$\frac{(x-c_{ix})^2}{r_{ix}^2} + \frac{(y-c_{iy})^2}{r_{iy}^2} + \frac{(z-c_{iz})^2}{r_{iz}^2} - 1 < 0 \quad i = 1,\ldots,m$$

where m is the number of threats. The calculation of threat intersections within a leg is similar to that outlined in section 2. The route cost function is also extended to include a penalty term for excessive climb and descent angles, similar to that already used for turning angles.

A minimum altitude penalty can also be added to the cost function (2) for the three-dimensional model. This is of the form $(z_{\min} - z)^2 * \omega$ where ω is a penalty parameter. When using DIRECT, the *initial* box sizes and waypoint selection can be made to exclude the possibility that z is negative. However, if DIRECT is run in restart mode, then, after the first cycle, it may be possible for a waypoint to be placed below the minimum height. Inclusion of a minimum altitude constraint guards against such an occurrence.

Tests with three–dimensional routes

The parameter settings used in all of the following tests are given below.
Maximum climb/descent angle = 50°; Maximum turn angle = 42.5°;
Minimum route separation distance = 200 metres;
Minimum waypoint altitude = 500 metres; Minimum leg length = 2km;
Threat penalty factors = 10; Turn penalty factors = 5;
Climb/descent angle penalty factor = 5; Minimum altitude penalty factor = 1000
Minimum route separation penalty factor = 10

The starting guesses given to DIRECT are straight-line solutions with all waypoints being given the minimum altitude setting of 500 metres.

In the first example, DIRECT is given the same initial box dimensions for altitude as for the horizontal directions. Subsequently, we examine the more realistic case where the box size in the *x*, *y* plane are greater than for the *z* axis.

In all the following tests we use the approach of obtaining routes for each aircraft separately. Where the waypoint addition procedure is used, we always start with a small number of waypoints and allow additions after every other restart.

Test 7.1

For this problem, which involves 18 threats, we compute routes for three aircraft, flying at the same speed. Each route uses three waypoints (initially placed equidistant along the straight line between the start and destination). The results found after running DIRECT for 2*64 iterations are shown in Figure 7.1.

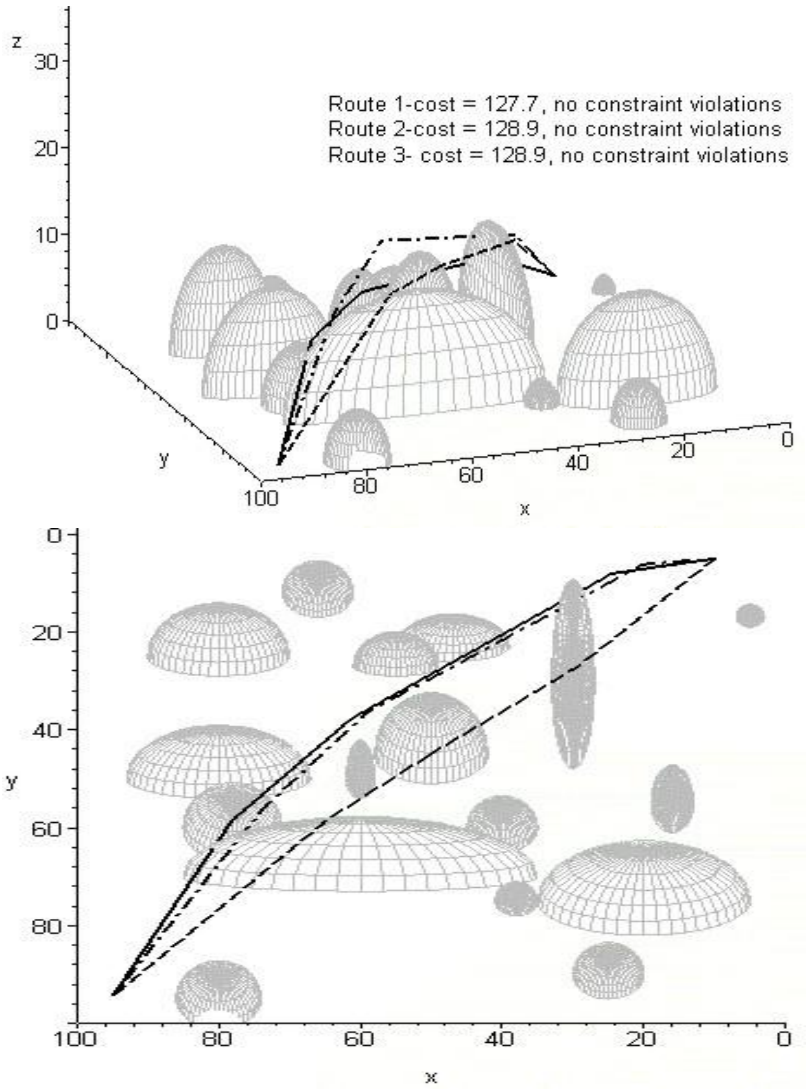

Figure 7.1

We can see that all three routes fly around the edge of the two large hills rather than flying directly over the peak. All three routes are distinct. Route 1 is the shortest (due to it being lower than others) and routes 2 and 3 are equal in length. A total of 27472 evaluations was used to find the above routes which took a little over five minutes.

In the above test, DIRECT was started with hyperbox edges that were the same in the vertical and horizontal directions and hence the solution routes are often flown at quite high altitudes. On real-life missions this may not be acceptable and the range for the *z*-component of the waypoint may then be smaller than that for the *x*- and *y*-components. In other words the waypoints could be given a freedom of say ± 15km horizontally but only ± 1km vertically.

The fact that the co-ordinates of each waypoint are now represented using two contrasting values (i.e. horizontal and vertical distance) will have implications for DIRECT. The reason for this is as follows. When any particular hyperbox has been selected for subdivision, it is the longest edge that is identified and trisected. Therefore, the horizontal waypoint coordinate dimensions will tend to be subdivided much more often than the vertical one. In order to prevent this, we consider an approach which was recommended by Jones et al. (1993). The aim is to transform and scale the problem so that all the optimization variables are centred on zero and have initial box side lengths of ± 1. This will mean that all the box dimensions will be equal in length and hence DIRECT should not pay too much attention to any particular variable(s). When passed in to the SRM cost function, the variables are then scaled up to their actual size and a route cost obtained in the usual way.

For the next example we make use of this *unit-box* size idea and also re-introduce the algorithm variants DIRECT-1 and DIRECT-2.

Test 7.2 (using unit-box scaling)

In this test we run the three forms of DIRECT on the same eighteen-threat problem as in Test 7.1. The initial box surrounding each waypoint has edges of ± 50km in the horizontal directions and ± 2km in the vertical direction. Restricting the z-dimension may mean that routes can no longer simply fly above high terrain and so we have used large horizontal side lengths to allow for a wider area of freedom of movement in the *(x,y)*-plane.

We allow DIRECT and DIRECT-1 to run for 4*64 iterations and DIRECT-2 to run for 4*192 iterations. The three waypoints are initially placed equidistant along the straight line joining the start and target points. The results are presented in Table 7.1.

Route	DIRECT		DIRECT-1		DIRECT-2	
	Route cost	Evaluations	Route cost	Evaluations	Route cost	Evaluations
1	163.0	52664	160.7	54360	172.3	37048
2	216.5		165.0		148.2	
3	169.5		216.4		152.1	

Table 7.1 – Solutions to Test 7.2

DIRECT-2 has overall produced the better routes in the fewest evaluations. DIRECT and DIRECT-1 have generally produced similar routes with DIRECT-1 being slightly more

expensive due to the periodic aggressive searches. The routes found by DIRECT-2 are shown below.

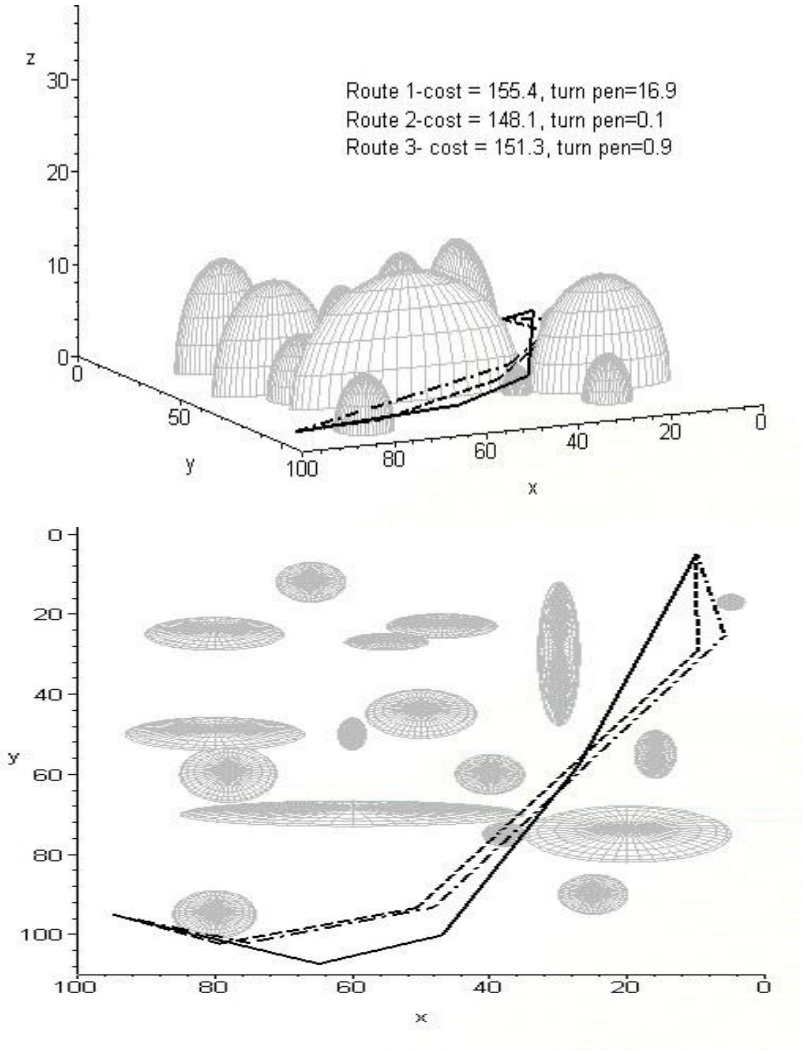

Figure 7.2

We can see from Figure 7.2 that the routes are flown much lower than those in Figure 7.1 and have therefore been forced to perform more horizontal manoeuvres. All three routes follow similar trajectories and fly over the small threat between the higher obstacles. As all three routes approach the target, sharp turns are made which has resulted in turn constraint penalties.

Test 7.3 (using waypoint additions)

We now run the three forms of DIRECT on the same problem, but using the waypoint-addition procedure. Each route starts off with two waypoints with the opportunity to add waypoints at the end of the first and third re-start. The results from this test are presented in Table 7.2.

Route	DIRECT		DIRECT-1		DIRECT-2	
	Route cost	Evaluations	Route cost	Evaluations	Route cost	Evaluations
1	135.8 (2)	43778	135.0 (2)	50670	179.3 (0)	33656
2	206.6 (0)		159.1 (1)		139.5 (1)	
3	482.4 (0)		163.0 (1)		460.1 (3)	

* The values in parenthesis show the number of added waypoints

Table 7.2 – Solutions to Test 7.3

DIRECT-1 has found the best routes (shown in Figure 7.3) which improve on those found with a fixed number of waypoints. DIRECT-1 is again the most expensive variant whilst DIRECT-2 produces quite good routes and is the cheapest method.

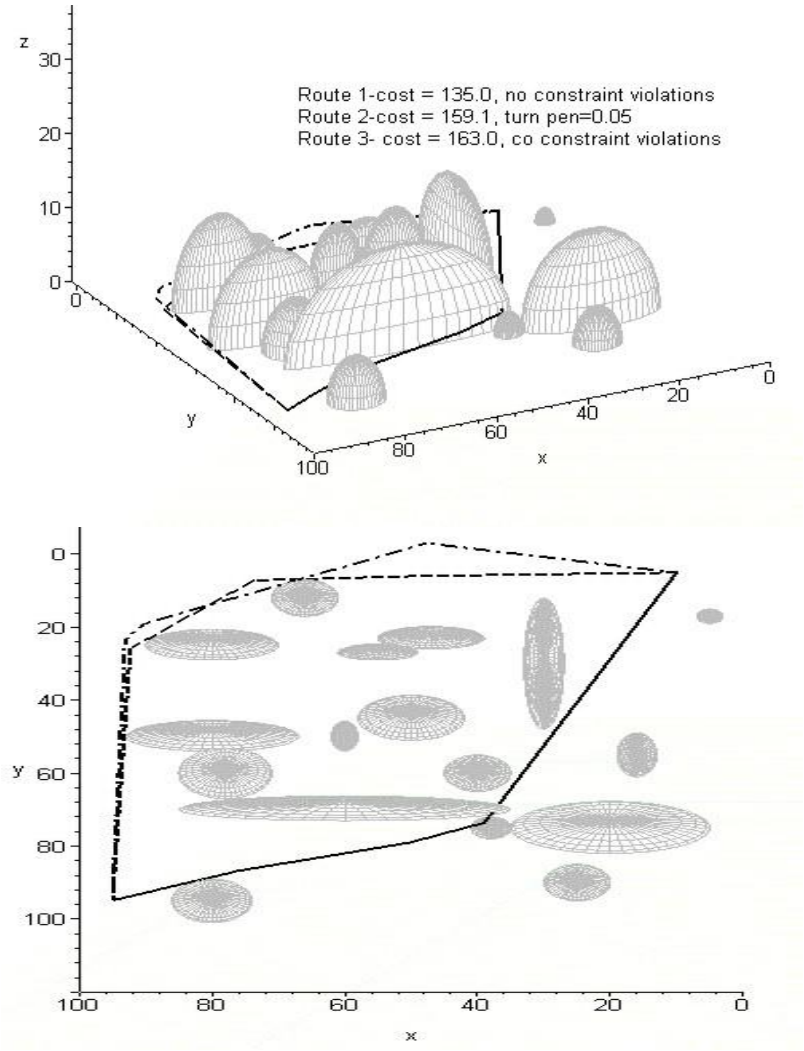

Figure 7.3

Test 7.4

We now consider some out-and-home routes over the same terrain and threats as used for tests 7.1 – 7.3. We start with a two-aircraft problem and allow both routes to use eight waypoints (including the fixed target point). We let DIRECT and DIRECT-1 run for 4*128 iterations and DIRECT-2 for 4*382 iterations. Due to the added complexity of the out-and-

home problem, we relax the restriction on height by increasing the box size for altitude to ± 4km. The results from this test are presented in Table 7.3.

Route	DIRECT		DIRECT-1		DIRECT-2	
	Route cost	Evaluations	Route cost	Evaluations	Route cost	Evaluations
1	6762.5	223276	1684.7	227366	6670.2	56384
2	8435.7		1575.2		6663.6	

Table 7.3 – Solutions to Test 7.4

In this case DIRECT-1 has produced the best routes (see Figure 7.4) but is again the more expensive variant (although not much worse than DIRECT). The routes found by DIRECT-2 are similar to those found by DIRECT-1 but the cost is higher because the turn constraints are violated more heavily, though a significant saving in the number of evaluations is observed. DIRECT found the worst routes and was only slightly more economical than DIRECT-1.

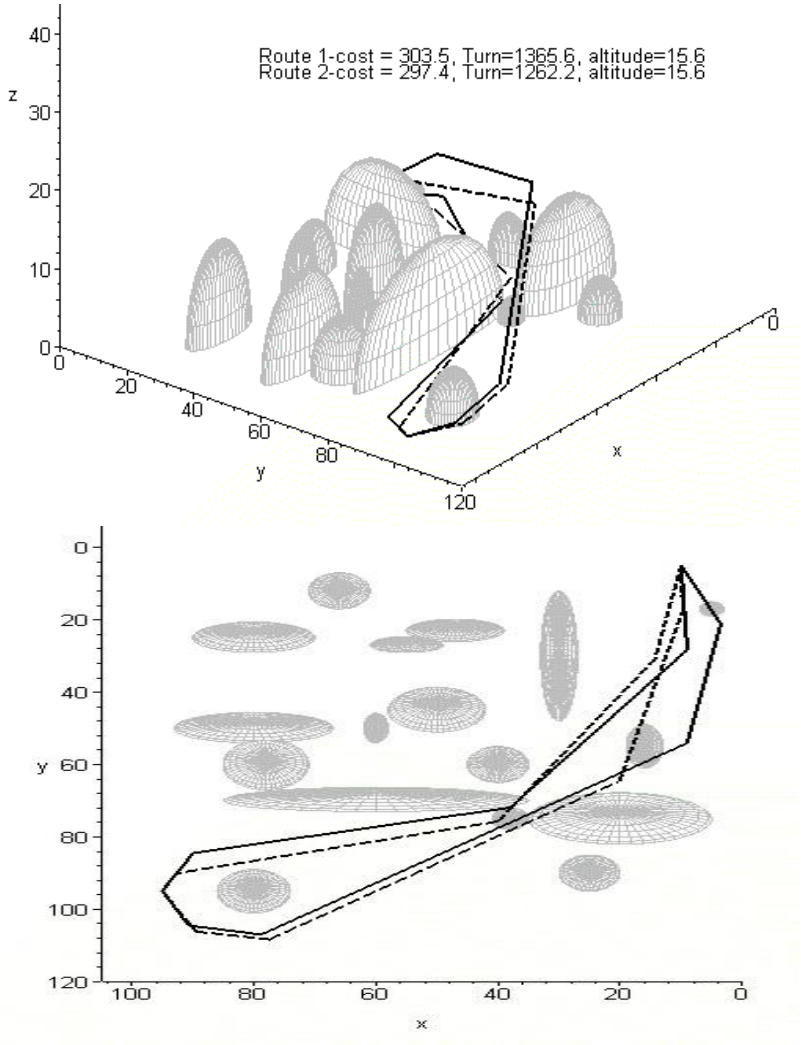

Figure 7.4

Both the outbound and return legs of each route follow the tight valley through the high terrain. The turn angle penalties occur near the transition from outbound to return route.

Test 7.5 (using waypoint additions)

For this test we apply the automatic waypoint addition procedure to the same problem as in Test 7.4. Each route starts off with three waypoints (including the fixed target points). We allow 4*128 iterations per route with the opportunity to add waypoints occurring at the end of the first and third re-start. The routes produced are shown in Table 7.4. In some cases we have obtained better solutions than when a fixed number of waypoints was used. DIRECT-2 and DIRECT-1 have found routes with fewer constraint violations with the former producing shorter routes in considerably fewer evaluations (though only a single waypoint was added to each route). DIRECT on this occasion is the most expensive version but produced poorer routes that violated threat and altitude constraint violations. Figure 7.5 shows the routes computed by DIRECT-2.

Route	DIRECT		DIRECT-1		DIRECT-2	
	Route cost	Evaluations	Route cost	Evaluations	Route cost	Evaluations
1	8254.4 (2)	118528	10455.1 (3)	113948	8878.9 (1)	35694
2	3204.7 (4)		321.8 (4)		9534.5 (1)	

The values in parenthesis show the number of added waypoints

Table 7.4 – Solutions to test 7.5

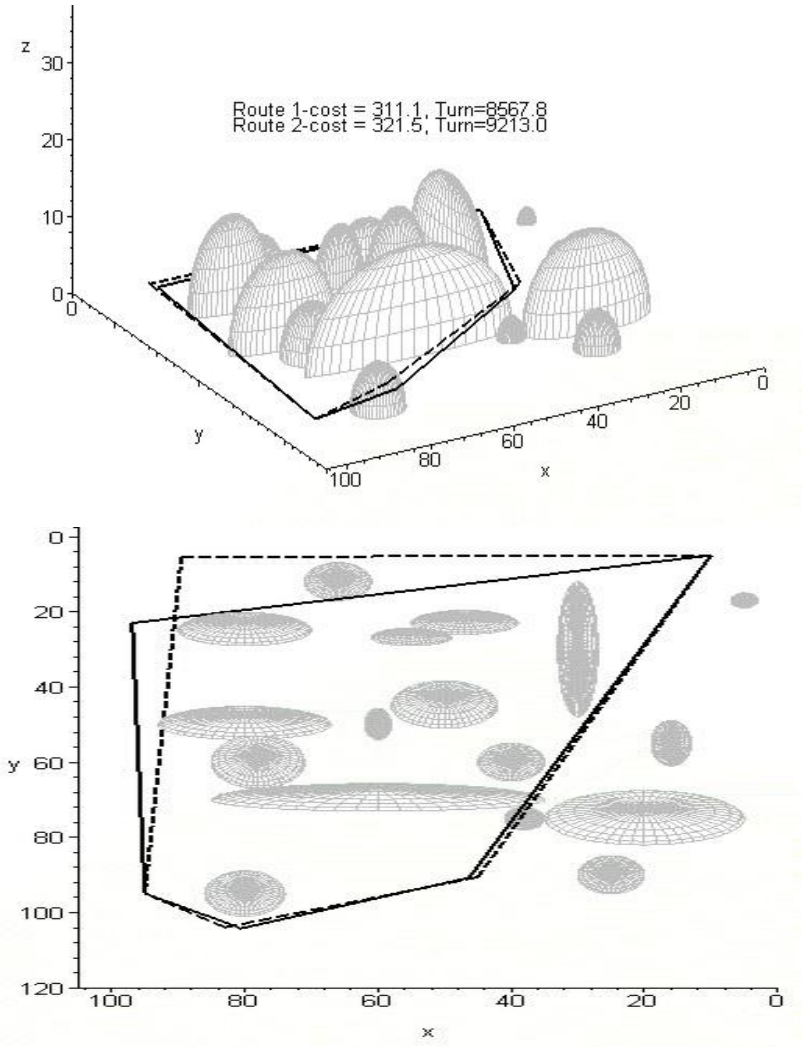

Figure 7.5

Both routes follow similar paths (unlike those computed by DIRECT and DIRECT-1). The outbound route legs fly around the outside of the threats and terrain on the left-hand side and the return legs cut through the valley between the high terrain.

8. Conclusions

The extension of the aircraft routing problem to handle several routes in both two and three dimensions has generally been handled well by DIRECT and its variants.

The problem of computing multiple routes was tackled in several different ways. We started by solving for all the routes simultaneously; but we found that posing the problem in this way meant that a large number of variables were used (particularly for out-and-home problems) and this made obtaining routes very expensive because many iterations were needed to ensure that a sufficiently refined search was performed. Jones (2001) remarks that DIRECT is at its best for problems in less than about 20 variables. We have sometimes dealt with routing problems where the number of waypoint coordinates exceeds thirty.

A second approach was to calculate outbound and return routes separately. We found that this approach delivered very poor solutions and, although it worked well on single-aircraft problems, it seemed unsuitable for multiple aircraft.

Our third approach was to solve for each route separately. This became the preferred strategy since good solutions were generally obtained more cheaply than with the other two approaches. Based on our experience so far, we have found that decomposing the problem into smaller ones improves both the accuracy and computational costs of the DIRECT algorithm. Currently we are investigating an extension to approach 3 which aims to break the problem down even further.

We have also experimented with a strategy that allows waypoints to be added "on the fly" during a solution. We found that starting with a small number of waypoints and adopting a rather cautious strategy for adding new ones could be both effective and economical. A strategy of only allowing waypoints to be added at specified intervals ensures that the problem size does not grow too rapidly.

We have found that periodic restarts of the DIRECT algorithm can often improve solutions, reduce computational costs and help convergence to a global optimum. These benefits may be due to the fact that, after a restart, waypoints will be centred upon "good" points and so in the subsequent iterations fewer boxes will be found potentially optimal. Therefore, fewer boxes will be selected for subdivision which in turn means that fewer boxes will have to be stored. This prevents evaluations from being wasted on large numbers of "bad" boxes which will help accelerate convergence to the global optimum and reduce the total number of evaluations used. There is scope for further investigation on how best to choose the re-start frequency.

Changes were also made within the framework of the DIRECT method. Two modified algorithms were tested (DIRECT-1 and DIRECT-2). The former uses periodic "aggressive

searches" whereby the best box of every size is subdivided after a predefined number of evaluations have passed without improvement to the route cost. It also incorporates stopping criteria which terminate the algorithm once the smallest box is less than a predefined value. DIRECT-2 has the same features as DIRECT-1 except that it performs only *one* subdivision of each potentially optimal hyperbox. Our experience has shown that DIRECT-1 often produces similar results to DIRECT although there were occasions when the aggressive searches led to significant improvements in solutions. DIRECT-2 was by far the most economical version and was often competitive with DIRECT and DIRECT-1. However, disappointingly it has performed less well when used with the RRM cost function. This is a matter for further investigation.

Three-dimensional routing problems where waypoints are expressed as (x, y, z) coordinates require the use of "unit boxes" in DIRECT. DIRECT only trisects along the longest edges; and since altitude coordinates will typically have smaller ranges than those in the x, y plane, it was observed that DIRECT did not explore these variables so thoroughly. A transformation of variables so that a unit box can be used around each waypoint removes any bias towards the longer box edges.

The ideas outlined in this paper are a contribution to the development of a complete routing solution system. Such a system should also automate, as far as possible, the selection of initial guessed solutions and the choice of box-size for each waypoint variable.

9. References

Hewitt, C., & Broatch, S.A. (1992). *A Tactical Navigation and Routeing System for Low Level Flight* (Tech. Rep.).Rochester, Kent: BAE SYSTEMS Avionics Ltd.

Hewitt, C., & Martin, P. (1998). *Advanced Mission Management* (Tech. Rep.).Rochester, Kent: BAE SYSTEMS Avionics Ltd.

Zhiqiang Chen Holle, A.T. Moret, B.M.E. Saia, J. and Boroujerdi (1995), A., Network routing models applied to aircraft routing problems, Simulation Conference Proceedings, 1995. Winter 1995, page(s): 1200-1206

Zabarankin, M, Uryasev S and Murphey R (2006), Aircraft Routing under the risk of detection, *Naval Research Logistics Vol 53, 8,* 728-747

Karczewski, III, Norbert J.(2007), Optimal Aircraft Routing in a Constrained Path-Dependent Environment , Master's thesis, Naval Postgraduate School Monterey CA

Carlyle W.M., Royset J.O., Wood, R.K.,(2007), Routing military aircraft with a constrained shortest-path algorithm, http://www.nps.navy.mil/orfacpag/resumePages/papers/woodrkpa/CarlyleRoysetWoodAircraftRouting.pdf

Bartholomew-Biggs, M. C., Parkhurst, S. C., & Wilson S. P. (2003). Global optimization approaches to an aircraft routing problem. *European Journal of Operational Research, 146,* 417-431.

Bartholomew-Biggs, M .C., Parkhurst, S. C., &Wilson S. P. (2003). Using DIRECT to Solve an Aircraft Routing Problem. *Computational Optimization and Applications, 21,* 311-323.

Jones, D. R., Perttunen, C. D., & Stuckman, B. E. (1993). Lipschitzian Optimization without the Lipschitz Constant. *Journal of Optimization Theory and Applications, 79,* 157-181.

Al-Sultan, K. S., & Al-Fawzan, M. A. (1997). A Tabu Search Hooke and Jeeves Algorithm for Unconstrained Optimization. *European Journal of Operational Research 103,* 198-208.

Chelouah, R., & Siarry, P. (2000). Tabu Search applied to global optimization. *European Journal of Operational Research, 123,* 256-270.

He, J., Watson, L. T., Ramakrishnan, N., Shaffer, C. A., Verstak, A., Jiang, J., Bae, K., & Tranter, W. H. (2002). Dynamic Data Structures for a Direct Search Algorithm. *Computational Optimization and Applications, 23(1),* 5-25.

Jones, D. R. (2001). The DIRECT Global Optimization Algorithm. *Encyclopedia of Optimization, vol. 1,* 431-440.

Watson, L. T., & Baker, C.A. (2001). A fully distributed parallel global search algorithm. *Engineering Computations, 18,* 155-169.

Wilson, S. P. (2003). *Aircraft Routing Using Nonlinear Global Optimization*. Unpublished PhD thesis, University of Hertfordshire.