Mission Planning For Synthetic Aperture Radar Surveillance

DAVID M PANTON Center for Industrial and Applicable Mathematics

University of South Australia

Mawson Lakes

South Australia 5095

ANITA W ELBERS P&O Nedlloyd

P.O. Box 966

3000 DM Rotterdam

The Netherlands

August 20, 1998

Abstract

The Microwave Radar Division of the Defence Sciences and Technology Organisation employs synthetic aperture radar to obtain high-resolution images of selected ground targets. It uses this device mounted on an aircraft, to scan up to 20 rectangular regions called swaths to obtain images with resolutions down to one meter. Missions consisting of a designated sequence of swaths and flight paths are planned using mapping software. Previously DSTO had been determining the best tours for missions by visually tracking possible swath sequences from a starting base to an ending base. This method was time consuming and did not guarantee optimality in terms of distance travelled. We developed optimization software tools to plan mission tours more efficiently. DSTO can now plan missions with up to 20 swaths in a few seconds, rather than the hour it took using the visual approach. Proposed tour lengths show an average improvement of 15 percent over those manually produced. The software incorporates methods for dealing with the operational problems of no-fly zones and shadowing associated with images.

The Microwave Radar Division (MRD) of the Defence Sciences and Technology Organisation (DSTO) in South Australia approached our center late in 1995 to discuss the possibility of optimizing, or at least automating, a mission-planning process associated with the use of a synthetic aperture radar (SAR) device used for aerial surveillance. The *Ingara* system developed by DSTO is capable of producing very-high-resolution images of ground targets. Apart from the obvious military and strategic usefulness of this system, SAR has the potential for a number of civilian applications, not the least of which is in aerial mapping. SAR carries out surveillance over rectangular strips called *swaths*. Swath widths are typically 24 kms. but may be narrower if higher resolution images are required. Swaths may be of any operationally realistic length. In carrying out the scanning process, the aircraft flies parallel to the swath length, typically at a distance of 27 kms. and at a height of 3048 meters (10,000 feet). It projects a radar beam continuously across the width of the swath, with the echoed beam returning to the aircraft for processing and image reconstruction. As with a light source, the radar beams create shadows when blocked by topological features within the swath region.

DSTO has been using this system for its military exercises in the Northern Territory Australia. In these situations, it needs a limited number of swath sites for surveillance. Indeed in some cases, it scans the same swath from each of its four sides to image a target of particular significance, although the mission would then count this swath site as four separate swaths. At present, the radar is mounted on a DC3 aircraft, which has a limited range, and missions are typically no more than 20 swaths in length; however, future use with longer range aircraft may demand longer missions of 50 swaths or more. Currently, the DSTO plans missions using the NAVTOOL mapping software it developed. NAVTOOL provides the aircraft navigator with detailed information on the location of swath coordinates (way points) within the mission region and other needed information. Included in this information is a prescribed sequence for scanning the swaths in the mission, together with the associated flight path. Mission planners within the DSTO have been using the mapping software to visually plan the best possible sequence. With missions in which swaths are widely separated this task is relatively straightforward. When swaths are tightly bunched and overlapping, however, this visual approach is much more difficult. Planning the best swath sequence can take up to an hour for a mission containing 15 to 20 swaths, using this manual approach. Nor did this method guarantee optimal sequences. DSTO wished to improve on this process with the view to extending the range of its missions and to enable it to reschedule missions in flight.

Our basic problem was to devise a method (preferably optimal) of planning missions containing up to 20 swaths more efficiently. In planning a mission, one determines the best sequence of swaths and associated tracks, starting from a given base, and ending possibly at some other specified base. The basic problem is clearly similar to the traveling-salesman problem (TSP) [Lawler et al 1986], but different because of the way swaths are scanned (Figure 1).

The aircraft track in the surveillance process is some distance to the left or the right of the swath center line (H to I). The aircraft may scan the swath by projecting a beam to its left or to its right, and it may fly in either direction (H-I or I-H). Thus there are four alternative tracks along which the aircraft may scan the swath. In some situations the track is specified or *mandatory*, but in general, one must consider all four possible tracks $A \to B \to C, D \to C \to B, AA \to BB \to CC$, and $DD \to CC \to BB$, when determining the optimal mission sequence. All points displayed on the tracks (such as A, B etc.) are referred to as way points. The section of a track such as $A \rightarrow B$ is called the prerecord distance and is used to align the aircraft in preparation for scanning between B and C. The location, length, and width of each swath is specified in each mission, together with the location of each way point in relation to the swath. Each swath is identified by four sets of nodes (A,C; D,B; AA,CC; DD,BB), each pair being a possible candidate for the aircraft track in an optimal solution. Once one selects the first member in a pair (for example A), then the second member (C) is automatically selected since the aircraft must fly along the only track that starts with that member ($A \rightarrow C$). Our problem can not be modeled as a TSP because we can select only one pair of nodes for each swath.

We were asked to include several additional tasks in solving the problem:

(1) To incorporate mandatory runs;

(2) To incorporate no-fly zones;

(3) To determine shadowing for each swath; and

(4) To determine strategies for aerial surveillance of an entire region, (regional surveillance).

Heuristics versus Integer Programming

In our initial model for generating optimal or good tours we wanted to obtain solutions in real operational time. The DSTO was prepared to accept good solutions that were not necessarily optimal to achieve this. Another factor influencing our initial model was our conception of the problem in terms of the swath geometry.

We decided to approach the problem by first viewing each swath in terms of its center-line coordinates H and I. Each swath is then associated with two nodes, but once we chose a particular node (for example I) then the associated node (in this case H) is automatically the next node in the tour. It is then relatively straightforward to adapt one of the tour construction algorithms to construct candidate tours from the given data. Recent work [Jünger, Reinelt, Rinaldi 1995] suggested that farthest insertion combined with a 3-opt procedure would give good results, and so we adapted these methods to generate initial tours from base to base using swath center-line coordinates. Once we determined the initial swath sequence, we chose the best track for each swath. We constructed a network that included all swath-track nodes connected according to the optimal TSP sequence. Thus for example, if we start at the base and first go to swath 6, our network would contain the edges joining the base to all nodes associated with swath 6. Then the next swath in the optimal TSP sequence might be swath 2, whereupon our network would contain all valid edges from nodes associated with swath 6 to nodes associated with swath 2, and so on. The resulting tour then visited each swath and selected the best track for each swath using a shortest path algorithm. We realized that this approach would not necessarily give optimal tours in general, since the tours would be influenced by the location of the tracks as much as the location of the swath center-lines; however, we conceived of the swaths as widely separated and the track lines as closer to their swaths than to other swaths. When the first of our data sets arrived, together with the NAVTOOL software, we found that our solutions were in fact *worse* than the manually derived results. A display of the swaths and their associated tracks on the mapping software quickly showed why. Rather than being widely separated, swaths were in fact close together and in many cases overlapping. Furthermore, the swath tracks were further from these swaths than the swaths were from each other. Our strategy of employing TSP and shortest path in combination was clearly not going to work. Faced with a deadline, we devised an alternative method using integer programming (IP). We had discussed this approach earlier but discarded it because we thought it would be too slow and not as easy to interface with to carry out other tasks associated with the problem. On the other hand, an IP model could give optimal solutions not guaranteed by a heuristic. We developed and refined an IP model in a few weeks (appendix) using the GAMS [Brooke et al 1996] software with CPLEX [1994] as the solver. The main difficulty in using an IP model for a problem of this nature is to ensure that it eliminates subtours from solutions. It is well known that even for problems of the size we faced in this project (up to 80 nodes), eliminating subtours requires a prohibitive number of constraints. Branch-and-cut methods for dynamically inserting subtour-elimination constraints requires resources that were not available at that time, and so we had to rely on a version of the Miller-Tucker-Zemlin (M-T-Z) constraints [Lawler et al 1986] used for eliminating subtours. It is well known that although these constraints are only $O(n^2)$ for an n node problem, they tend to cause difficulties in the LP solution process. Despite this we hoped to obtain acceptable solution times given the small size of the problems. For the first two data sets provided by DSTO, flight 9 containing nine swaths, and flight 12 containing 15 swaths, we obtained acceptable solution times of around three or four minutes. Subsequently, we used an aggregated version of the subtour elimination constraints (appendix) to reduce these times significantly (Table 1).

Data set	Optimal tour (swath sequence)	Optimal distance (nautical miles)	Manual distance (nautical miles)	Time(seconds)
Flight 9	5,8,7,6,9, 4,3,2,1	346.2	411.3	2.1
Flight 12	9,11,15,6, 3,4,2,1, 14,7,8,10, 5,12,13	214.2	253.3	8.8

Table 1: The lengths of the optimal tours for two flights improve on the manually calculated best distances. The execution times are for a SUN Sparc 10. Flight 9 is a nine swath problem, and flight 12 is a 15-swath problem. Both missions begin and end at the same base.

When we displayed these solutions on the NAVTOOL mapping software, some nonintuitive tours appeared that had not been considered as options in the manual construction process.

Geographical Separation

While we had expected that missions with 20 swaths would take longer to solve than those with 15, the first data set containing 20 swaths gave an interesting and surprising result: it proved intractable. This data set contained two widely separated sets of swaths. In this mission, the aircraft scanned 10 swaths in the vicinity of the Darwin region, flew south, and scanned a further 10 swaths in the Katherine region. Even when we extracted certain swaths from each region, leaving as few as five in each, the problem was still intractable or took several hours to solve. We tried a second data set, containing three geographically separated swath sets and observed the same phenomena. Finally we tried a third data set in which 20 swaths were randomly distributed within the same regional boundary (Darwin to Katherine). In this case, we found an optimal solution in 107 seconds, clearly indicating that increasing the number of swaths did not cause the intractability. It appeared that the nature of the data- the geographical separation of swath sets- was making the problem difficult to solve. Adjustments to CPLEX run-time parameters and attempts at either aggregation or disaggregation of constraint sets within the model had no influence on the outcome. On the other hand, we found that when we added constraints that linked the swath sets with paths between track end points, we obtained acceptable solution times. One solution then was to devise a heuristic procedure to allow the mission planner to intervene in the solution process for missions with geographically separated swath sets. We wrote software that allowed the mission planner to identify from NAVTOOL appropriate connecting edges which were then written into GAMS in constraint form. While this method provided good solutions, in retrospect, we realized that the need to manually intervene and uncertainty about what constituted 'geographical separation' reduced the effectiveness of the overall process.

Problems with geographically separated data sets prompted us to investigate a more refined approach to handling subtours. A number of relaxation-based methods exist for dealing with subtours. One such method, the subtour relaxation, relaxes both the subtour constraints and the integer requirement. One detects subtours by solving the subtour separation problem using such methods as the Gomory-Hu algorithm [Lawler et al 1986] and disconnected subtours by using a simple depth-first search. As one progressively detects subtours, constraints are added that remove them from the model until one finds an optimal tour containing no subcycles. This process can be automated using software such as that in the CPLEX callable libraries. In practice, it is found that one needs to add few subtour constraints to obtain an optimal solution.

The so-called subtour separation method cannot be used in this case, however, since it is not possible to relax the integrality condition. It is the integrality condition which ensures that a single node in each swath set is selected; once this condition is dropped the unit flow in each swath may be shared between more than one swath node, complicating the application of the standard separation procedure. Nevertheless, we thought we might make some gains by simply dropping the M-T-Z constraints and solving the problem as an IP. We could detect disconnected (integer) subtours and insert constraints prohibiting them. This approach proved effective. We could now solve to optimality in acceptable times problems containing geographically separated swath sets that were originally intractable. This method also proved effective in solving standard problems, so we decided to use it as a general solution technique. Table 2 shows the times for solving both the large and spaced 20 data sets, together with revised times for flight 9 and flight 12.

Mandatory Runs

In certain situations, the mission planner must fix the way in which an individual swath is to be scanned, for example when an initial run shows too much shadowing over a region of interest within the swath. In that case, the mission planner specifies that the swath must be scanned from the other side, or in an extreme case, from all four sides. When scanning from all four sides is required, it is simply a matter of reproducing the swath at the same location, but with tracks taken parallel to its width in addition to

Data set	Time (secs.)	Optimal solution	Best manual distance	Distance using heuristic
large	24.5	644.4 nm	679.6 nm	654.8 nm
spaced 20	14.5	447.5 nm	637.3 nm	475.7 nm
Flight 9	0.6	346.2	411.3	-
Flight 12	3.6	214.2	253.3	-

Table 2: We improved times and distances for all data sets by using CPLEX callable library routines.

those parallel to its length. To prescribe mandatory runs, one can

(1) Select the radar used (left or right), or

(2) Select the side for aircraft tracking.

Option 1 allows the user to specify either left or right radar (for example if the aircraft has only left radar available, one would specify left radar for all swaths). In option 2 one chooses the side with reference to the given $H \to I$ direction in the data-input file. Once one has made all selections, one writes an include file containing the necessary constraints and reads it into the GAMS code. Table 3 shows the result of running flight 12 data with swath 1 scanned using right radar and right side (r/r).

Data set	Mandatory runs	New (old) tour length	Time (secs.)
Flight 12	1:(r/r) 5:(l/r) 13:(l/r) 15:(l/l)	234.0 (214.2)	7.1

Table 3: For Flight 12, we made mandatory runs in which swath 1 is scanned with right radar and on the right side; swath 5 with left radar and on the right side; swath 13 with left radar and on the right side; and swath 15 with both left radar and left side. The length of the tour in this case, is as expected, longer than for the case with no mandatory runs.

No-Fly Zones

The region in which the swaths to be scanned are located may include areas that cannot be flown over (no-fly zones). Aircraft may pass through any corner or fly along any edge of a no-fly zone, however. No-fly zones are specified by a sequence of up to eight sets of coordinates, which collectively define a convex polygon with up to eight edges. Several no-fly zones may be specified in one region. Each no-fly zone may be defined as a convex set by representing each edge as a linear constraint of the form $p * lat + q * long \leq r$ where lat and $long$ are coordinates of the corners of the no-fly zone. Inclusion of a point within the no-fly zone will then require that the coordinates of that point satisfy **all** constraints.

One must distinguish between a **track** and a **path** in the problem network. The aircraft flies a **track** along a swath as it obtains radar images. It flies a **path** between two track end-points for two distinct swaths. Any track that violates a no-fly zone must be excluded from any solution; however, potential paths between swaths that violate no-fly zones need be considered only if they occur in the optimal solution.

Since each problem has a large number of potential tracks and paths and a large number of no-fly zones we use an initial screening algorithm to identify those tracks and paths that will never violate the no-fly zones. For each no-fly zone, we define an enclosing rectangle (Figure 2). Any track or path whose end points are **both** inside one of the regions, A, B, C, or D, will not violate the no-fly zone (for example, tracks 1 and 2); however tracks with one end point in one region and the other in another region (for example tracks 3, 4, and 5) may violate the no-fly zone, and we cannot determine whether they do by looking at their end points. Nevertheless this initial screening process identifies many tracks or paths of type 1 and 2.

We then consider the remaining tracks or paths that **may** violate the no-fly zones. Clearly any track or path with both end points inside the no-fly zone violates the no-fly zone. For the remaining tracks or paths neither of whose end points lie inside the no-fly zone we compute their intersections with all edges of the no-fly zone. Those that do not intersect with the edges, do not violate the no-fly zone. Those that do intersect one or two edges of the no-fly zone, do violate the no-fly zone. Some special cases that could arise in practice require rules.

We ruled that paths or tracks whose **end points** touch the boundary of the no-fly zone violate it.

We ruled that paths or tracks that touch a corner of the no-fly zone do not violate it.

We ruled that paths or tracks that lie along a boundary edge of the no-fly zone do not violate it.

Once we have accounted for the violated *tracks* in the GAMS code , we run the swath model and find the optimal solution without initially allowing for violated *paths*. At this stage, we test each path that connects track end points for a no-fly zone violation. When we detect a path from one swath to another that crosses a no-fly zone, we construct a network (Figure 3). The starting vertex in the network is the end point node of one track, and the ending vertex is the start point node of the other track. The intermediate vertices are the corners of the no-fly zone(s) violated. (A path connecting two tracks may violate more than one no-fly zone). We define potential paths as network arcs only if they do not cross no-fly zones. We find the diversionary path between tracks for the two swaths by finding the shortest path between one swath track's ending node and the other's starting node. This path will travel along the edges of the no-fly zones but not pass through them. We then write information about this solution to the output file by interleaving the coordinates for the path around the no-fly zone between the coordinates for the swath tracks created in the standard output. The NAVTOOL mapping software uses this additional information to display the diversionary path. When a no-fly zone lies between two or more swaths, it may be possible to find an alternative solution (suboptimal in the original solution), that avoids the no-fly zone and is shorter than the path found by the method discussed above. One can use a simple strategy in this situation. Once one defines the no-fly zone one can run the model a second time with the paths and tracks that violate the no-fly zone excluded. One can then compare the solutions and use the one that is best.

Shadowing

An area is in *shadow* if the radar cannot illuminate it because of a difference in elevation between adjacent areas (Figure 4). If we scan such a swath from a flight track along one edge, parts of the swath will lie in the shadow of mountains or hills. If too much of any swath lies in shadow, the aircraft must fly another track that causes less shadowing. To calculate the amount of shadowing in a swath we divide the swath into *pixels* (square areas of one second of latitude at the equator or 30.87 meters). Away from the equator we reduce this dimension by $cos(\phi)$, assuming that ϕ is the average latitude of the swath. We number the pixels according to which track we use to image the swath, (Figure 5). The position of pixel (i, j) on the swath can be determined by reference to an origin which is located at the corner of the swath where scanning commences.

One can compute the average elevation of the surface in each pixel and define the average height at the center of each pixel - the height line (i, j) of pixel (i, j). Figure 4 shows a 10 by 20 pixel swath, where each pixel is divided into four parts to indicate its center. To calculate the shadowing of the whole swath, we calculate the shadowing of each strip. Suppose we fly the track along the $x = 0$ line in figure 4 and we observe pixel strip i. We first create a cross section ((x, z)-plot) on the y-coordinate of the center of the pixels in pixel strip i and calculate the average height of all pixels (Figure 6).

A *gradient line* (j) is the line defined by the (x, z) coordinates of the airplane, the average (z-coordinate) of pixel (i, j) and the x-coordinate of the center of the pixel. Starting with the first pixel in strip i, we calculate the gradient line (1). If the z-coordinate of the intersection between gradient line (1) and height line (2) (int(1,2)) is bigger than the average elevation of pixel 2 (h2), then pixel (i,2) is in shadow, otherwise the height of pixel (i,1) does not cause any shadow on pixel (i,2). In our example, the second pixel is not in shadow. If pixel (i,2) is in shadow, we compute the intersection between gradient line (1) and height line (3) and determine whether pixel 3 is in shadow. We stop when the z-coordinate of the intersection is smaller than zero or when we find a pixel that is not in shadow. We repeat this process for all the gradient lines until we reach the end of the strip.

An example of a pixel in shadow is pixel 8. The intersection between gradient line (7) and height line (8) is greater than the average height of pixel 8, and we continue with calculating intersections between gradient line (7) and height line (9). We stop at pixel 16 because int(7,16) is smaller than h16. Thus we find pixels 8 to 15 to be in shadow.

When using this method we will never find the first pixel of each strip in shadow. In the approach we adopted we in essence assume that if the pixel on the swath edge shadows the second pixel then it is likely that the unobserved pixel on its other side shadows it. So for these specific pixels, we determine the gradient of the line defined by the elevations of pixel 1 and 2 (which we call a). If this gradient has a different sign than the gradient of gradient line (1) (call it b), the first pixel is not in shadow. If the signs of a and b are the same, the first pixel is in shadow only if the absolute value of a is smaller than the absolute value of b (Figure 7). Figure 8 shows the calculations of shadowing based on scanning the swath shown in Figure 4 from each of its four sides.

Shadowing will exist for any flight path. We can reduce the shadowing by scanning the swath more than once using different paths and combining the images. Figure 9 shows the amount of shadowing that remains when information from two paths is combined.

Regional Surveillance

DSTO was also interested in the feasibility of devising a systematic method of planning a mission to scan a complete geographical region. This task is clearly different from scanning specified swaths. A possible commercial application for such regional surveillance (RS) is in crop-dusting. The regional-surveillance (RS) problem adds another difficult element to the mission planning problem: deciding how to cover the region. One approach to this problem would be to construct a cover for the region consisting of a suitable set of swaths, and to then apply the methods discussed above to find the optimal tour. An alternative approach would be to devise a global model which determines both the cover and the optimal tour simultaneously. We are currently investigating algorithms for solving this problem.

Conclusions

The tools we developed for planning SAR surveillance missions have improved operational efficiency. We have reduced the time needed to plan missions, including the time needed to deal with situations such as mandatory runs, no-fly zones and shadowing, significantly. On average, the total mission distances found using the optimization techniques are 15 percent shorter than those found manually. The GAMS/CPLEX optimization tools proved to be a very good environment in which to develop these procedures because they allow easy interface with the NAVTOOL mapping software and with other code necessary to control menu options and to write include files.

Future missions, regional surveillance, and possible commercial applications will almost certainly involve a larger number of swaths, which will test the effectiveness of the optimization model. Despite the removal of the M-T-Z constraints, we believe that it may be necessary to speed up the solution process, suggesting the need for other facet-defining constraints, an adaptation of the subtour-separation method, or perhaps an alternative formulation.

Appendix

Let s be the number of swaths and n the number of nodes, including the starting and ending base. Then the total number of nodes in the resultant network is $n = 4s + 2$. Taking aircraft track directions into account, four tracks for each swath are possible. Each is associated with two nodes, giving a total of eight nodes. However, a particular pair of track nodes, for example DD,BB, is associated with a given direction and hence can be merged into a single node representing that direction. As a result, four distinct nodes (i) can be associated with each swath (Figure 1). This reduction is important to the size of the resultant model. A swath set $S_k \;\; k = 1 \ldots s$ is a subset of the node set $N = \{1, 2, \ldots, n\}$. Thus $S_1 = \{2, 3, 4, 5\}$ etc. Let x_{ij} be a 0-1 variable indicating whether or not the edge from $i \rightarrow j$ is used in the optimal tour, and let w_{ij} be the length of edge $i \rightarrow j$. Note that when computing the length of edge $i \rightarrow j$, we use the distance from the final way point (i) of one swath to the starting way point of another swath, *plus* the length of the second swath track. Then our problem is to minimize the length of the tour

$$\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \; x_{ij}$$

subject to the constraints

$$\sum_{j \neq 1,n} x_{1j} \;\; = \;\; 1 \tag{1}$$

$$\sum_{i \neq 1,n} x_{in} = 1 \tag{2}$$

$$\sum_{j \in S_k} \sum_{i \notin S_k, i \neq n} x_{ij} = 1 \quad \forall k \tag{3}$$

$$\sum_{i \notin S_k, i \neq n} x_{ij} - \sum_{l \notin S_k, l \neq 1} x_{jl} = 0 \quad \forall j, \quad \forall k \tag{4}$$

$$\text{no subtours allowed} \tag{5}$$

Constraint (1) requires that only one edge comes from node 1 (starting base), while (2) requires that only one edge goes to node n (ending base). Constraint (3) ensures that exactly one node in a swath set is selected (for example node 3 in $S_1 = \{2, 3, 4, 5\}$), while constraint (4) ensures that once a node in a swath set is selected, that node must connect to a node in another swath set (or node n). Constraint set (5) is the key source of difficulty with TSP models and models of this type. Unless it is included, subtours may occur.

There are several possible versions of the subtour-elimination constraints. However, most of these are not useful, because the number of constraints increases exponentially in the number of swaths. For instance, the number of the following subtour elimination constraints will be of order 2^s

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1$$

for every proper, nonempty subset Q of $S = \{1, \ldots, s\}$, where $|Q|$ is equal to the number of elements in the subset Q. An alternative set of constraints due to Miller-Tucker-Zemlin is given by

$$u_i - u_j + s * x_{ij} \leq s - 1 \quad 2 \leq i \neq j \leq s$$

where $u_i \geq 0$ are real variables.

The number of these constraints is only of order s^2; however, we introduce $2(s-1)$ new variables, which increase the solution time, and constraints of this type are known to cause difficulties in solving the associated LP. In the radar-mission-planning problem, we need only prevent subtours between swaths, which results in the following aggregated version of the above constraint set:

$$\sum_{i \in S_l} u_i - \sum_{j \in S_m} u_j + s * \sum_{i \in S_l, j \in S_m} x_{ij} \leq s - 1 \quad 2 \leq l \neq m \leq s.$$

Acknowledgements

We acknowledge the contributions to this project made by Jemery Day and Tania Marwood at the University of South Australia. In addition, we thank Bruce McDonald and Marshall Muller at DSTO for their support.

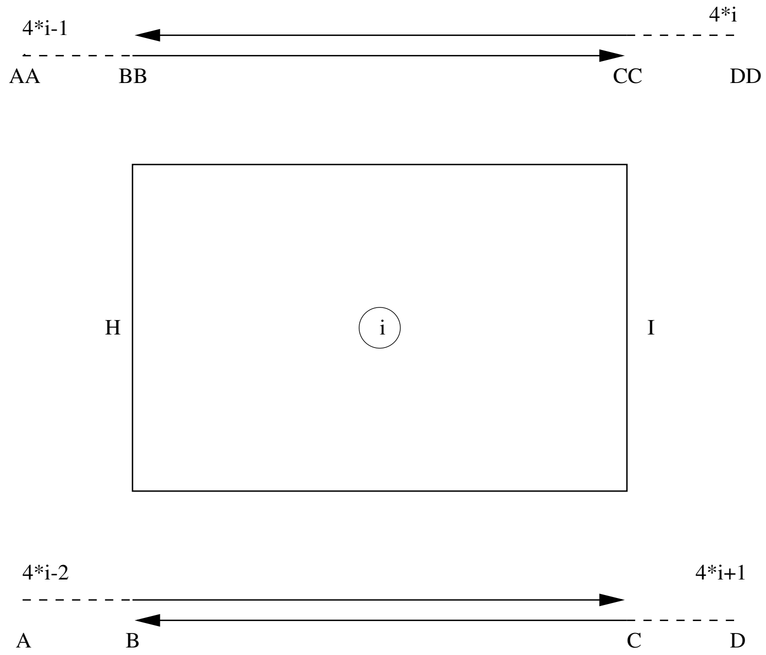
References

Brooke, A.; Kendrick, D.;and Meeraus, A.,"GAMS Release 2.25 A User's Guide", 1996, GAMS Development Corporation, Washington DC.

CPLEX, CPLEX Optimization, Inc. 1994.

Jünger, M.;Reinelt, G.;and Rinaldi, G. 1995, "The Traveling Salesman Problem" in *Handbooks in Operations Research and Management Science Vol. 7 Network Models*, eds. M.O. Ball et al, North-Holland, The Netherlands.

Lawler, E. L.;Lenstra, J. K.;Rinnooy Kan, A. H. G.;and Shmoys, D. B. 1986, "The Traveling Salesman Problem", Wiley, Great Britain.



4*i-1
4*i
AA
BB
CC
DD
H
i
I
4*i-2
4*i+1
A
B
C
D

Figure 1: Swath geometry. Four possible scanning tracks are shown as solid lines with arrows. Prerecord tracks are shown as dotted lines. Prerecord tracks allow the aircraft to setup prior to scanning the swath. The geographical location of the swath is defined by its center line coordinates H and I and the swath width. The geographical location of the tracks is defined by the way points shown at A, B, C, D, AA, BB, CC, and DD. Identification of nodes A, AA, D, and DD for swath i is determined by reference points $4 * i - 2$ to $4 * i + 1$. Node 1 is reserved for the base.

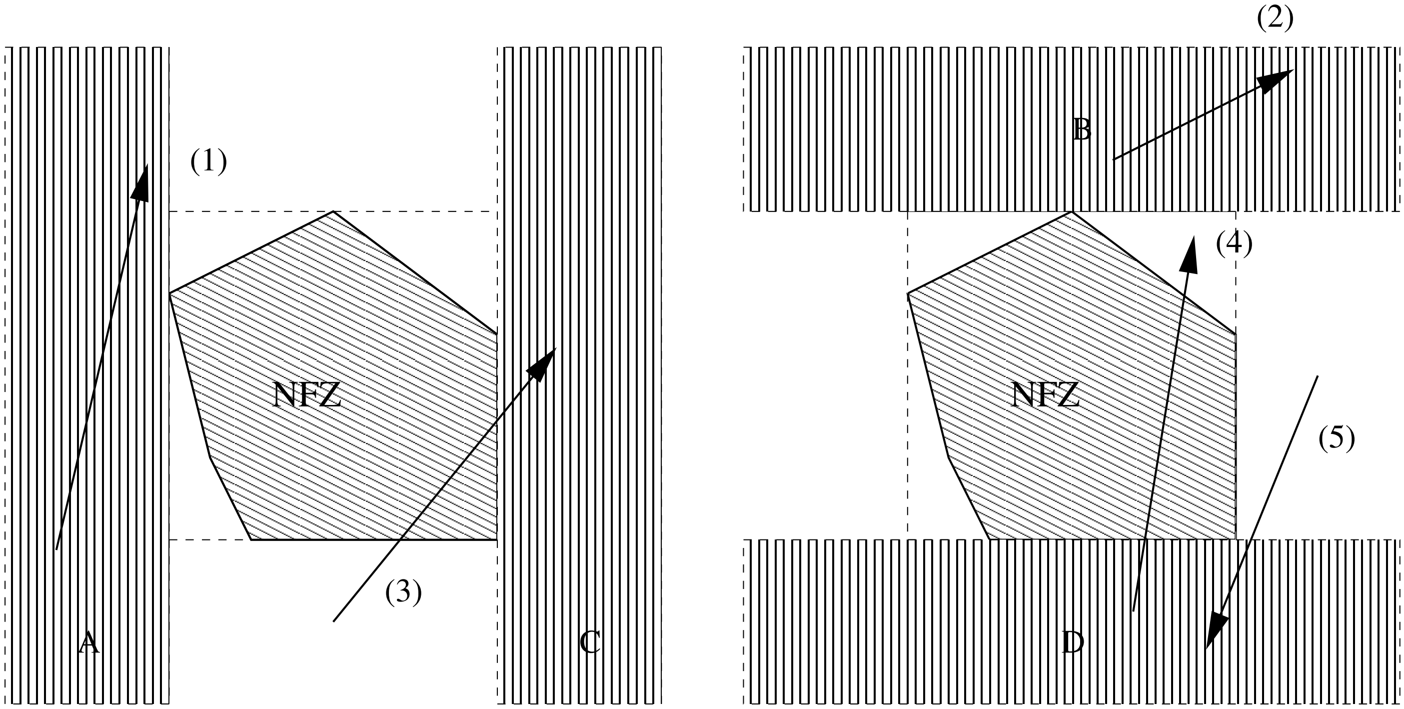
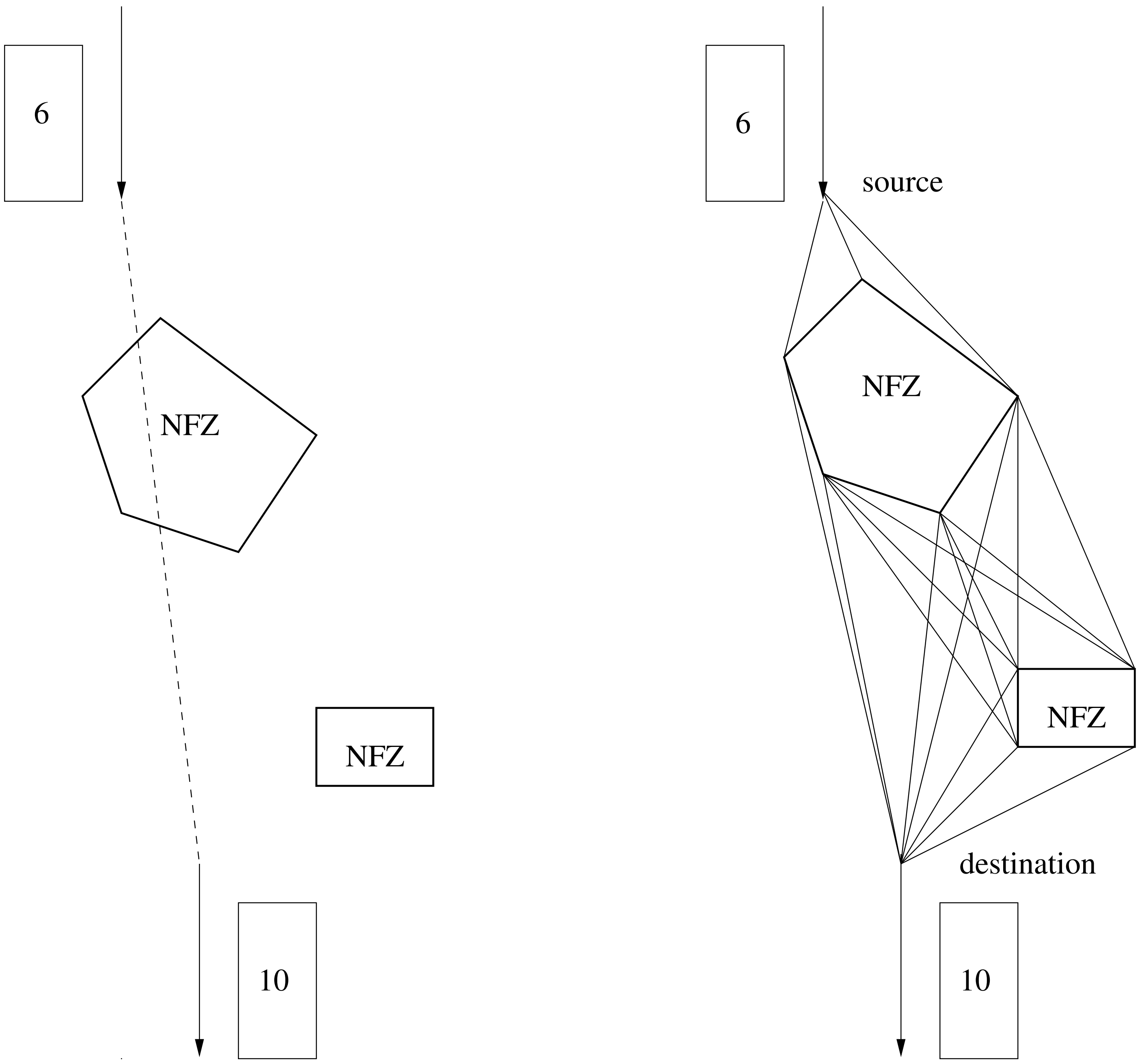
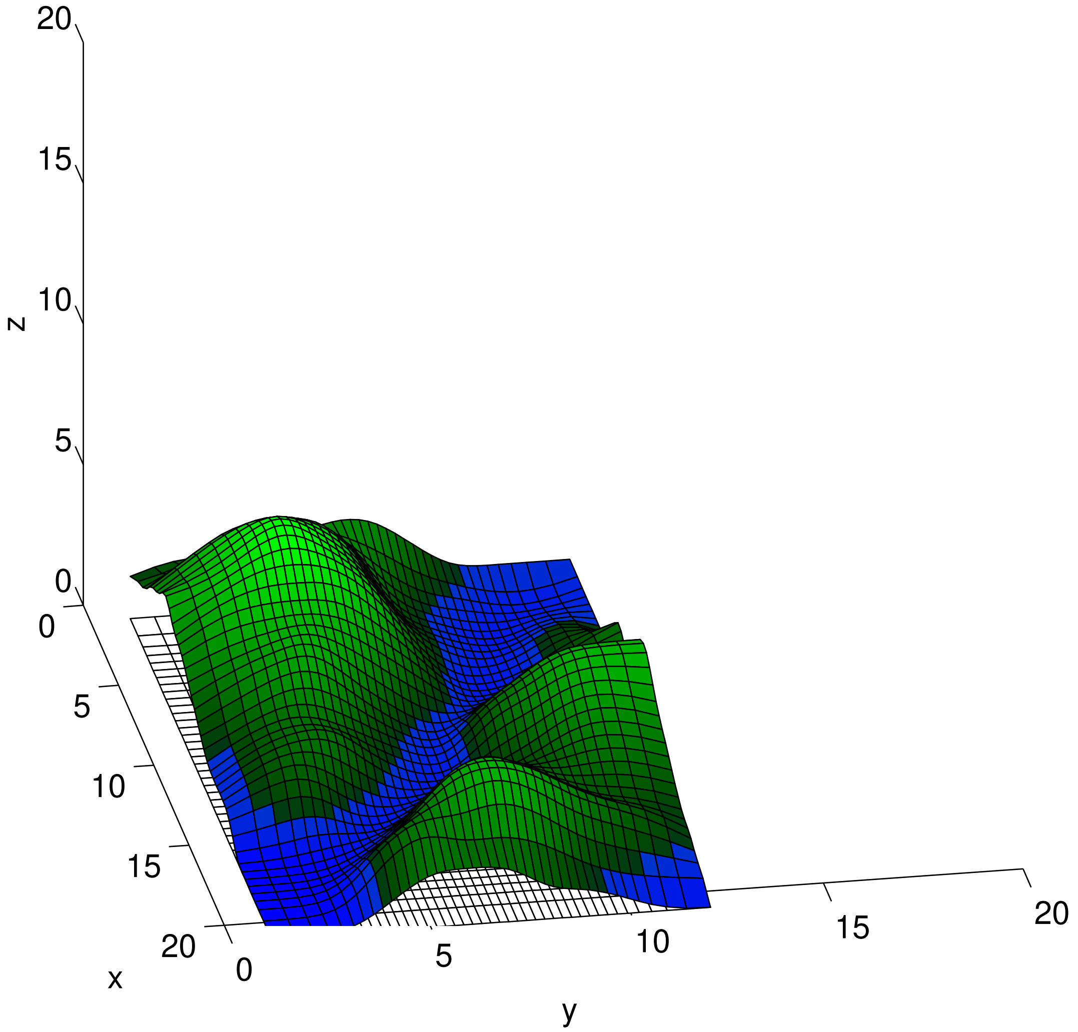

(1)
NFZ
(3)
A
C
(2)
B
(4)
NFZ
(5)
D

Figure 2: A rectangle enclosing a no-fly zone is used to identify tracks or paths which do not violate no-fly zones. Any track or path whose two end points lie in one region A, B, C, or D cannot violate the no-fly zone, for example tracks 1 and 2. Tracks 3, 4, and 5, with end points in two regions may or may not violate the no-fly zone.



(a) (b)

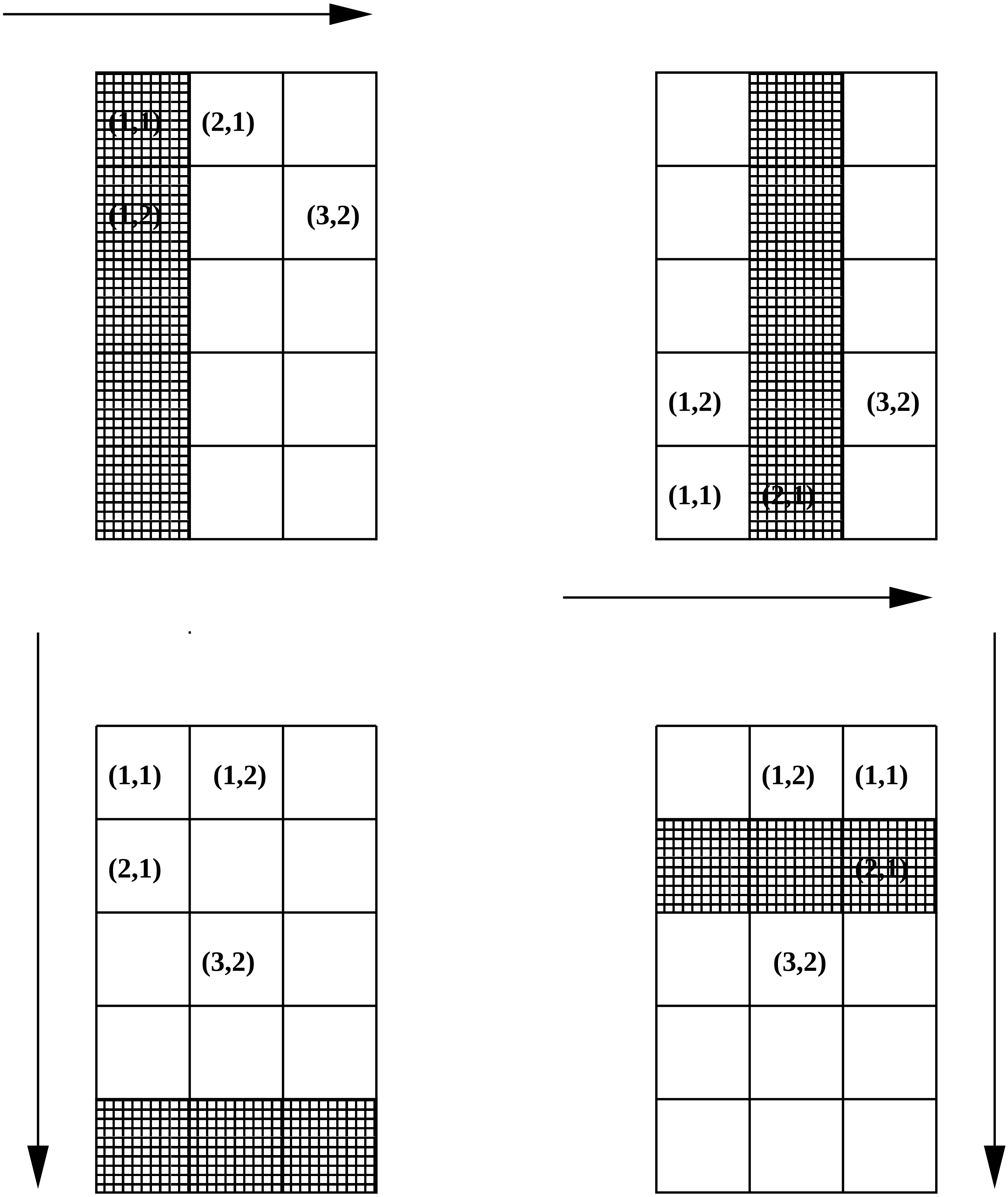
Figure 3: No-fly zone avoidance. When a path between the end of a track for one swath (for example swath 6) and the start of a track for another swath (for example swath 10) passes over a no-fly zone (dotted line in (a)) avoiding action must be taken. Figure 3(b) shows a network created using vertices consisting of the corners of the no-fly zones and with source and destination nodes the track end and track start points. Note that arcs in the network that would have cut across no-fly zones have been excluded. A shortest path problem is solved to find the best track from swath 6 to swath 10.

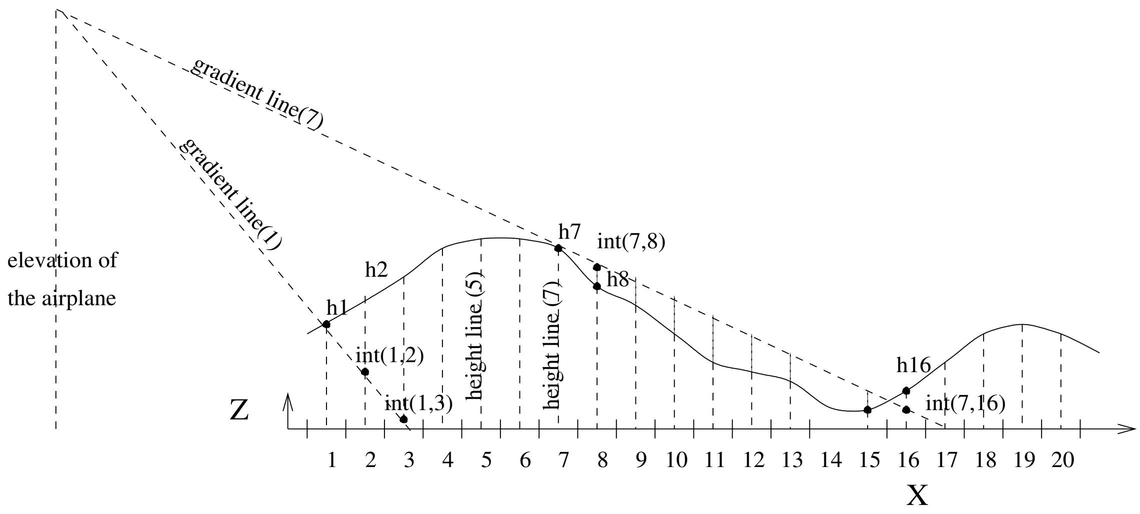


20
15
10
5
0
z
0
5
10
15
20
x
0
5
10
15
20
y

Figure 4: An example of a swath with elevation differences. Shadowing will occur in this case for each of the four possible flight tracks.

Figure 5: Pixel numbering. Pixels are numbered according to the flight track chosen. We take the origin at the corner of the swath nearest to the commencement of the flight track. Pixel strips at right angles to the flight path are numbered consecutively from this point. Pixels on each strip are numbered consecutively across the swath. Thus pixel (i, j) is located in strip i and is the j^{th} pixel from the flight track. Examples of pixel numbering for four different flight paths are shown.


(1,1)
(2,1)
(1,2)
(3,2)
(1,2)
(3,2)
(1,1)
(2,1)
(1,1)
(1,2)
(2,1)
(3,2)
(1,2)
(1,1)
(2,1)
(3,2)



gradient line(7)
gradient line(1)
elevation of
the airplane
h7
int(7,8)
h8
h2
h1
height line (5)
height line (7)
int(1,2)
int(1,3)
h16
int(7,16)
Z
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
X

Figure 6: A cross section of the region in figure 4 for a representative pixel strip i. hj denotes the average elevation of pixel j on this strip at its center. Height line(j) is drawn from the center of pixel j to intersect the surface at hj. We calculate if pixel j shadows pixel $j+1$ as follows. Consider gradient line (1) from the airplane which intersects the surface at h1. Since the z coordinate of the intersection between gradient line 1 and height line 2 (int(1,2)) is smaller than the average elevation of pixel 2 then pixel 2 is not in shadow. However, since int(7,8) which is the intersection of gradient line 7 and height line 8 is larger than h8, then pixel 8 is in shadow. This process is followed across each pixel strip.

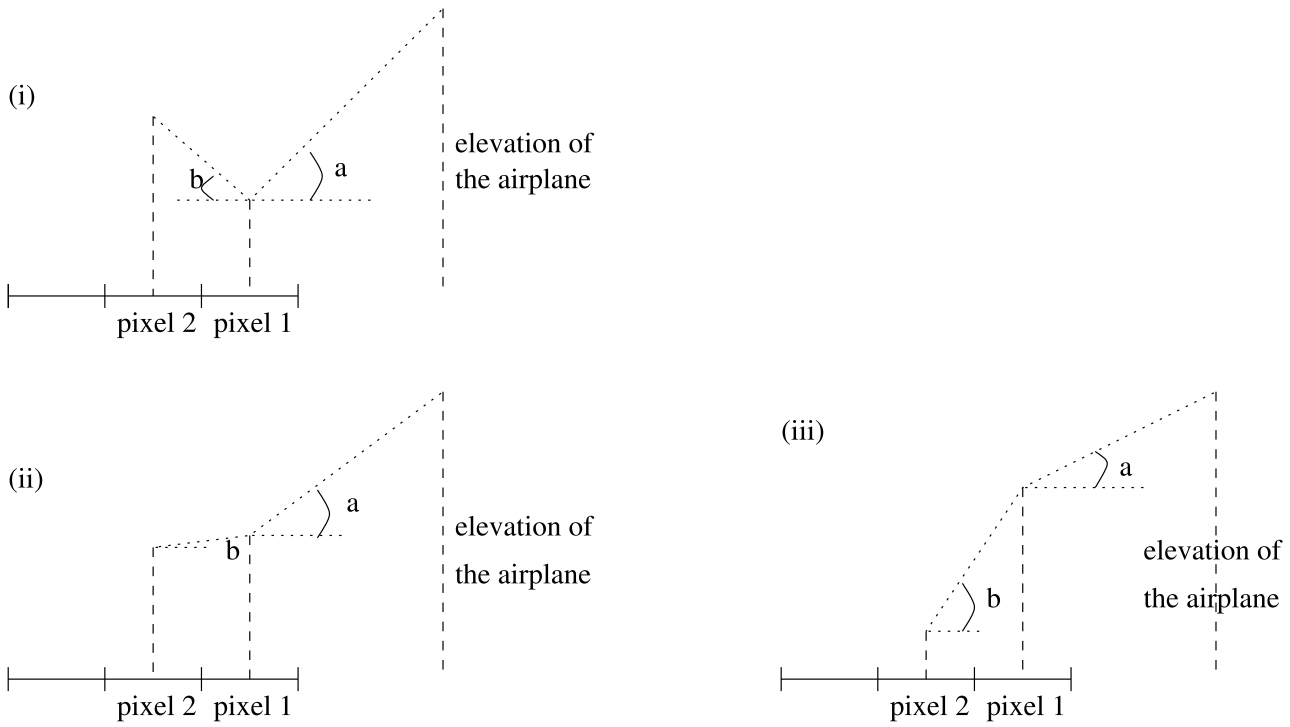
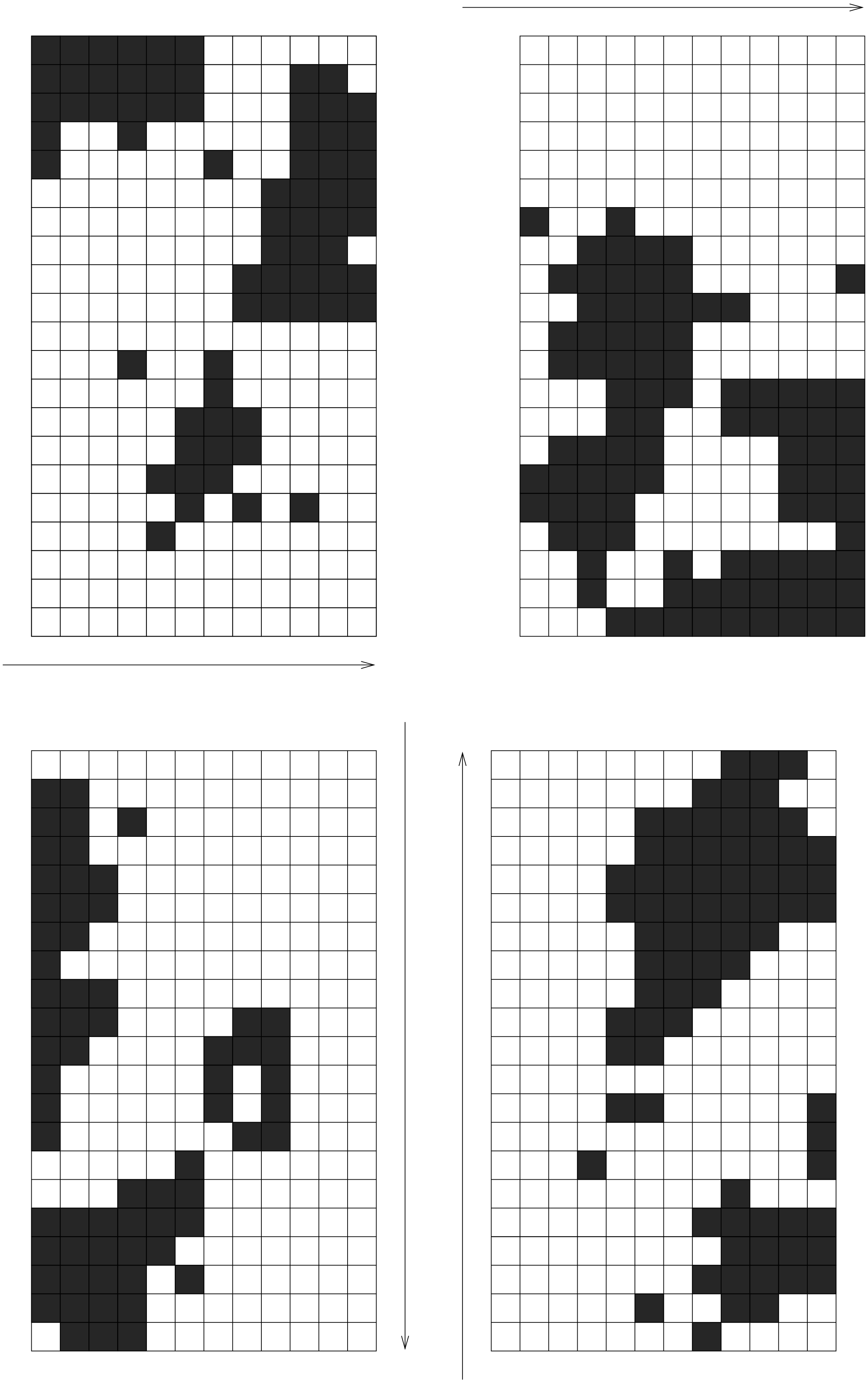
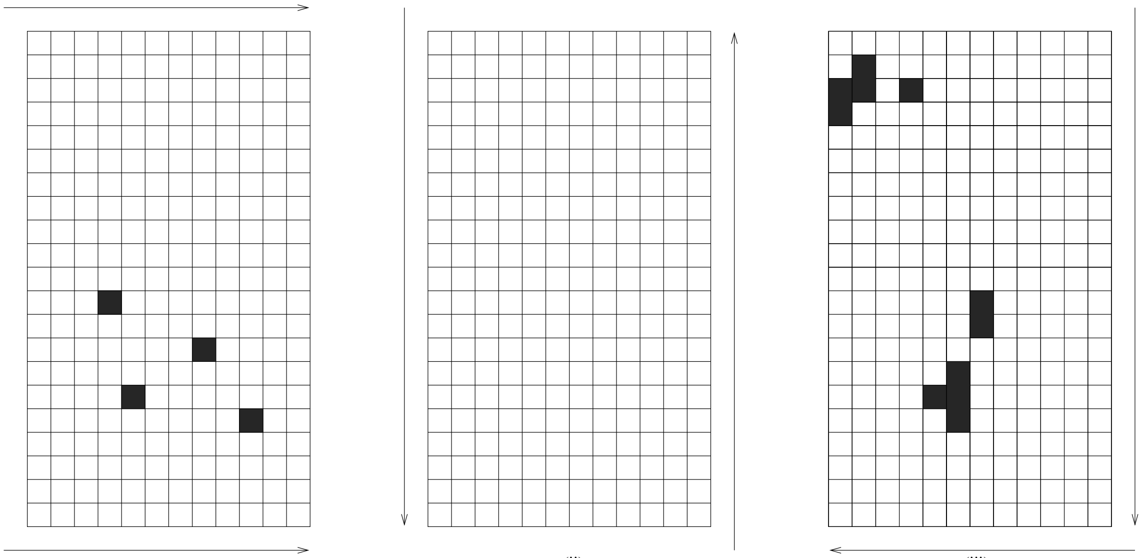

(i)
a
b
elevation of
the airplane
pixel 2
pixel 1
(ii)
a
b
elevation of
the airplane
pixel 2
pixel 1
(iii)
a
b
elevation of
the airplane
pixel 2
pixel 1

Figure 7: Three scenarios can be considered to see if the first pixel in each strip is in shadow: (i) In which a and b have different signs, and pixel 1 is not in shadow; (ii) in which a and b have the same sign, abs(b)$<$abs(a) and pixel 1 is not in shadow; and (iii) in which a and b have the same sign, abs(b) $\geq$ abs(a) and pixel 1 is in shadow.

Figure 8: Shadowing results for the swath displayed in figure 4 when this swath is scanned from each of its four sides.





(i) (ii) (iii)

Figure 9: Shadowing can sometimes be eliminated when a swath is scanned along more than one track and results are combined. Three examples for the swath in figure 4 are shown. In (i) scanning is done at each end; in (ii) along each side; and in (iii) along one end and one side. Only in case (ii) is shadowing completely removed when results are combined.