AUTOMATIC ROUTING MODULE

Janice A. Malin
Systems Control Technology, Inc.
1801 Page Mill Road
Palo Alto, California 94303

ABSTRACT

Automatic Routing Module (ARM), a tool to partially automate Air Launched Cruise Missile (ALCM) routing, is installed at HQ-SAC and is used operationally by JSTPS mission planners. For any accessible launch point/target pair, ARM creates flyable routes that, within the fidelity of the models, are optimal in terms of threat avoidance, clobber avoidance, and adherence to vehicle and planning constraints.

Although highly algorithmic, ARM is an expert system in the sense that it: (1) rapidly creates plans based on heuristics, or rules, supplied by planning experts; (2) relieves planners of much of the tedious and time-consuming portions of the route planning process; (3) supports both expert and non-expert planners in route creation; (4) allows the planner to change the rule base; (5) recommends a course of action and provides the planner with a means to modify that recommendation; (6) provides a menu-driven, user friendly interface plus interactive graphics; and (7) relies on a statespace, paths, and decision tree that must be searched, complete with cost function, to arrive at an optimal route.

Because of the heuristics applied, ARM-generated routes closely resemble manually-generated routes in routine cases. In more complex cases, ARM's ability to accumulate and assess threat danger in three-dimensions and trade that danger off with the probability of ground clobber results in the safest path around or through difficult areas. The tools available prior to ARM did not provide the planner with enough information or present it in such a way that ensured he would select this safest path.

1. INTRODUCTION

Systems Control Technology (SCT) began working on basic research in optimization applied to automated planning systems in 1978 under DARPA sponsorship. The motivation for this work was to:

1. reduce planning time and manpower requirements,
2. improve planning effectiveness,
3. produce timely responses to scenario changes, and
4. use existing technologies to solve the problem.

These efforts resulted in a basic approach to planning and a demonstration model referred to as AUTOPATH. AUTOPATH, as outlined in Figure 1, has broad applicability to a variety of mission planning problems. It has already been successfully used in cruise missile routing, force level planning, unit level planning, asset allocation, and an on-board processing experiment.

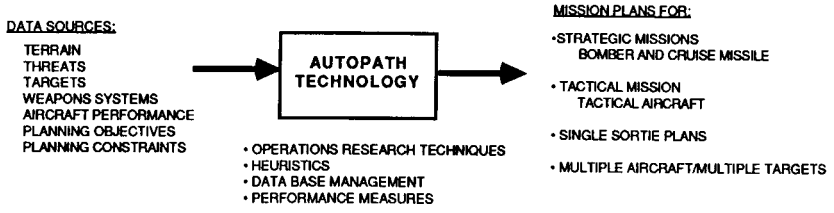

Figure 1 The AUTOPATH Technology Operates on Multiple Data Sources with Sophisticated Mathematical Algorithms and Optimization Techniques to Generate a Variety of High Level Mission Plans

This paper focuses on SCT's experience in using this basic technology to develop an automated Air Launched Cruise Missile (ALCM) routing tool for the Joint Strategic Target Planning Staff (JSTPS) at Offutt AFB. The JSTPS is colocated with Headquarters, Strategic Air Command (HQ SAC). The AUTOPATH research and development efforts laid the groundwork for a prototype system that was installed at HQ SAC in 1984. The Automatic Routing Module (ARM) that grew out of the prototype was installed in 1985. The transformation from research and development to an operational system was a complete success; ARM is used operationally at HQ-SAC to aid in the ALCM planning portion of the SIOP. ARM is a major software module. This paper is intended to briefly highlight certain aspects of the ARM software and development experience rather than to cover any aspect of ARM in detail. In that light, the topics covered are an overview of the project and its goals; the basic technology used to solve the problem; the implementation of the solution; and lessons learned from the project.

2. ARM OVERVIEW

The ALCM planning problem has many complex parts, all of which are labor and/or computer intensive processes. Computer processing plays a major role in the planning, primarily in creating and managing data bases and in analyzing and evaluating missions that were generated by expert human planners. SCT set out to automate the mission generation aspect of planning beginning in 1983.

2.1 The Planning Problem

The mission generation planning problem is to create a flyable route between a launch point and a target. This problem is easily stated but not easily solved. The ALCM is a long-range, terrain-following missile. It must follow a programmed path that guides it from launch to the target while ensuring that it does not run into any high ground (i.e., clobber). Using the terrain following capability, the missile's path must also avoid flying into enemy defenses or risk being destroyed before reaching the target. Avoiding clobber and enemy defenses along a single flight path is a difficult task that is compounded by a growing navigation error as the missile progresses to the target. To keep this error under control, the Inertial Navigation System requires periodic updates from a finite number of highly surveyed areas of terrain known as TERCOMs. Therefore, the final route must take advantage of the available TERCOMs and still avoid ground clobber and enemy defenses.

A human planner uses detailed maps which include the defenses and TERCOMs to perform this routing task. The planner has to determine the best path through the defenses and the best altitude for each leg of the flight. This is a gross oversimplication of the task because of many other considerations and constraints, some of which will be discussed later. The planner's task becomes very difficult in heavily defended areas and is further complicated by the number of missions to be planned.

Once satisfied with a mission, the planner provides the mission's flight plan to very detailed evaluation software which uses accurate vehicle performance models, local terrain, and detailed threat models to determine how good the path really is. The software can make some modifications to the path, but generally flags errors for the planner to rethink, fix, and resubmit the route for analysis. Because of the fidelity of the models, this software has been slow, making good candidate routes on the first pass highly desirable.

The objective of the ARM program is to create good quality candidate routes and to do so in a timely manner, i.e., less than ten seconds each. By meeting this objective, ARM relieved the mission planners of much of the tedious and time-consuming portions of the route planning process. It freed them to spend more time on difficult routes that do not follow all the rules given to the automated system, to fine tune routes, or to work other problems.

ARM can be given one or many missions, described by launch point/target pairs, to plan. For each mission, ARM creates a flyable route that is optimal in terms of threat avoidance, clobber avoidance, and adherence to routing constraints. Routing constraints include vehicle performance characteristics and heuristics, or rules of thumb, that a mission planner would apply if the route were to be created manually. Some of these rules are given in Table I.

Because of the heuristics applied, ARM-generated routes closely resemble manually-generated routes in routine cases. Manually created routes and ARM generated routes may diverge in more complex cases. ARM's use of three-dimensional threat models allows the system to rapidly determine the safest path around or through areas of high danger. This ability to use a deterministic approach rather than to rely on the human eye and brain to sort through the myriad of possibilties results in a route of equal or better quality than a manually generated route. Obviously, these complex cases are the ones for which ARM saves the most human planner time.

Table I ARM Rule Base Examples

VEHICLE CONSTRAINTS	PLANNING RULES	PLANNING HEURISTICS
• climb/dive rates	• terrain avoidance	• minimum fuel reserve
• turn radius	• threat avoidance	• minimum/maximum number of TERCOMS per route
• speed	• avoidance areas	
• maximum fuel load		• minimum/maximum distance between route segmentation points
• Warhead Arming Maneuver		
• Inertial Navigation System		• minimum acceptable target damage
• update frequency		• route dispersion
		• target avoidance
		• maximum distance between TERCOMs
		• maximum accumulated danger

2.2 Man-in-the-Loop

While ARM is a success, automation in mission planning has not reached the stage where the man-in-the-loop is not required. As one would expect, ARM requires a considerable amount of external data. Target, launch point, TERCOM, and defense data are just some of the data that must be provided and verified. Nominally this is the job of one person, the Mission Controller/Data Base Manager. In addition, all of the rules must be provided. These are saved from session to session, but can be changed as desired.

ARM provides the capability to review the routes and all intermediate information. The planner module is completely interactive and supports full, high-resolution, color graphics displays of the scenario, including latitude/longitude grid, launch points, targets, TERCOMs, and defenses. One very useful display feature that is provided and that is missing from planning maps is altitude-dependent danger contours for clobber danger, enemy defenses, or combined danger plus total danger contours at the optimal altitude for each statespace cell. With these displays, the planner has a much better understanding of why ARM chose a certain threat penetration, for example.

The graphics module also allows the planner to rapidly change the recommended route and reevaluate it. The planner has full control over changes to TERCOM selection, navigation point placement, leg-by-leg altitude and turn radius selection, as well as launch point and target specification.

2.3 ARM As An Expert System

ARM is not a true expert system as that term has come to be accepted today. ARM is an expert system in the sense that it:

1. creates plans based on rules supplied by planning experts,
2. supports the planner in his task,
3. allows the planner to change its rules,
4. recommends a course of action and provides the planner with a means to modify that recommendation,
5. provides a user-friendly, menu-driven interface plus a graphics display to support the planning process and understanding of the problem and its solution, and
6. relies on a statespace, paths, and decision tree that must be searched, complete with cost function, to arrive at an optimal route.

ARM is not a standard expert system in the sense that it is written in FORTRAN 77 rather than in a symbolic language, is highly algorithmic, and is data and I/O intensive.

2.4 Summary

The ARM system is largely a parameter or data driven system that provides mission planners with an effective tool for generating good quality candidate routes to be input to the detailed evaluation

programs. ARM further provides the capability to support real-time, rapid-strike route generation as well as studies of proposed vehicle modifications, routing logic alternatives, threat analyses, modeling alternatives, and various "what-if" hypotheses.

3. ARM TECHNOLOGY

The SCT AUTOPATH approach is based on decomposing the overall problem into several smaller, manageable pieces. The intent is to define functionally self-contained modules that are computationally practical.

Figure 2 provides a first level decomposition of the SCT AUTOPATH mission planning approach. The five main steps pictured are described in more detail below, as they relate to the cruise missile case.

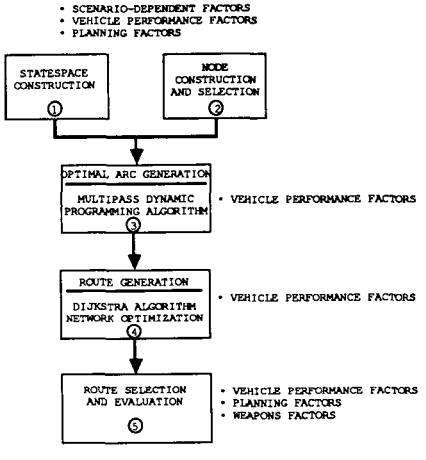

Figure 2 ARM Incorporates Many Military Planning Factors

Step 1: Statespace Construction: The statespace reflects a three-dimensional quantization of the geographic area of interest. The geographic area is divided into latitude-longitude cells with discrete altitude levels. Associated with each cell, and with each of the eight basic directions of traveling through the cell, is the 'cost' of traversing the cell. This cost is based primarily on the danger from enemy defenses; however, any number of other cost factors can be considered. For the cruise missile, it is important to consider the probability of the missile running into the ground. Thus, ARM also uses models that predict 'ground clobber' probability as a function of the vehicle's altitude and the terrain roughness. The terrain roughness is extracted from Digital Terrain Elevation Data (DTED). Danger from the enemy defenses are incorporated into the statespace through the cross-range and down-range threat models (or threat 'templates'). These templates are used to estimate the danger to the vehicle as a function of the distance and orientation from a particular threat type. An example of such a template is given in Figure 3.

Step 2: 'Node' Construction: In most mission planning applications, the vehicle has a number of routing constraints that have to be met. Most of these can be modeled by prescribing various geographic points, or nodes, through which (or through some of which) the vehicle must travel. In ARM, the nodes are launch points, TERCOMs, and targets. A complete mission can be represented by a sequence of nodes. The objective of the node construction step is to define a network that describes all possible pairs of nodes between which an ALCM could potentially travel in the given scenario. It is important to minimize the size of the this network. Thus, as many constraints as possible are considered. For example, if navigation updates are required every 'x' miles, the connections longer than this are not included in the network. Quotas of accessible nodes are also effective. These constraints are either hard constraints dictated by the vehicle or logical heuristics that are key to planning for a specific system. These constraints are defined during the 'knowledge acquisition' or requirements analysis phase of the effort.

Step 3: Route Segment Generation: The purpose of this step is to compute optimum route segments, or arcs, between each of the node pairs in the node network. For each node pair, one node is the origin node and the other is the destination node. For each destination node, a multipass dynamic programming algorithm (MDPA), is executed on a subset of the statespace that contains the node pair. The MDPA computes an optimum set of controls (directions of travel) for each cell in the statespace to the

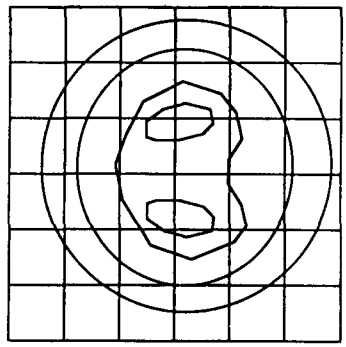

Figure 3 Sample Threat Template for a Left-to-Right Heading

destination node. Following the MDPA, the route retrieval algorithm is used to extract the optimum route segment that connects the origin node to the destination node. The route retrieval algorithms also perform whatever route smoothing is required. Once the route segment has been extracted, the total transit cost between the nodes is known and is stored for further processing.

Step 4: Route Generation: The next step in the process is to link these route segments into nearly complete routes, or paths. To preserve flexibility and to reduce the number of paths, the launch points and targets are not linked to the routes during this step. The basic algorithm used is the Dijkstra shortest path algorithm which provides a very efficient way to extract the set of route segments that produces a minimum cost route between an initial and terminal TERCOM node pair. Since each initial TERCOM can ultimately reach many other TERCOMs, the Dijkstra algorithm actually results in a tree for each initial TERCOM containing the best paths from the TERCOM to every other TERCOM it can possibly reach. The transit cost for each path of each tree is known and preserved.

Step 5: Route Selection and Evaluation: In ARM, the starting nodes in the Dijkstra search set are TERCOMs accessible to launch points and the terminal nodes are TERCOMs accessible to targets. The number of paths that can connect a launch point/target pair is now a manageable number with the transit cost of each readily available. Working backward from the target, it is a simple matter to select the tree and path that, together with the cost of traveling from the launch point to the tree and from the tree to the target, optimizes either the probability of arrival (Pa) or the probability of Damage (Pd) for the launch point/target pair. In doing so, ARM evaluates all potential routes it investigates against the planning criteria, but saves only the route description and evaluaton for the route it selects. Figure 4 shows a very simplistic example of one tree and its accessible launch points and targets.

This metnod of attaching the launch points and targets as the last step provides the capability to quickly link different launch points or targets to a validated path.

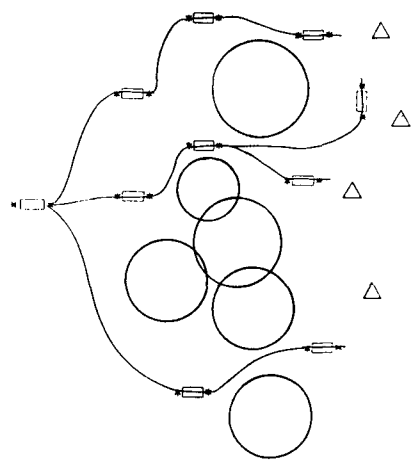

Figure 4 Sample Route Selection Options

4. IMPLEMENTATION CONSIDERATIONS

Because of the rule base and decision logic used in ARM, one may expect that it is written in a symbolic language. It is not. ARM is implemented in FORTRAN 77 and installed on an IBM 3090-200 and VAX 11/780. The IBM is the primary host with the VAX used for demonstration and off-site development and maintenance. FORTRAN was chosen for a variety of reasons and, in retrospect, appears to have been a reasonable choice. The driving factors were:

- highly algorithmic problem solution,
- manipulation of up to two gigabytes of data,
- required computational and I/O speed,
- limited virtual memory on the target IBM host computer necessitating overlays or task swapping,
- lower development cost using the FORTRAN prototype as a baseline, and
- the available compilers on the host computer at the time.

All of these factors precluded selecting a symbolic language. This list of constraints is still valid four years after the initial choice was made. Certainly FORTRAN is not the only answer for the near future, and technology is rapidly advancing; however, a caution must be raised here. While demonstrations written in symbolic languages may appear to solve the problem at hand, constraints such as those listed above should be evaluated before launching into the development of an operational automated system written in a symbolic language.

Once the language was chosen, core space, allocation of disk space, computational speed, and I/O speed became the critical design factors. As configured, the host computer provides only about seven megabytes of virtual memory. This is an absolute physical constraint while all the others were practical requirements. The virtual memory constraint was satisfied by using code overlays, increased disk I/O and, therefore, execution time, and a small loss of modeling fidelity, e.g., larger statespace quantization.

Obviously, the solutions to the core space problem compounded the other design problems. Execution time for arc and tree generation became an issue because of the increased I/O and the sheer volume of data to be processed. There were a limited number of disk packs available, and we desired not to have a single file span two disk packs. If run unrestricted, ARM would generate arcs between each pair of nodes and create a tree for every TERCOM. This could result in hundreds of thousands of arcs and many more trees than necessary. These problems were mostly solved by the heuristics. For instance, minimum/maximum TERCOM separation is considered plus a quota system for determining the number of arcs that will be generated for a TERCOM. Trees need only be built for those TERCOMs accessible to a launch point. Therefore, algorithms used in the prototype had to be redesigned to be more time efficient.

A commercial data base management system might have made the design easier, but at least at that time, would not have provided the speed needed. The data base management system is written entirely in FORTRAN and is adapted to ARM's needs. Many linked lists are used rather than wasting disk space for fixed format files. With the larger, faster disks available now, that same decision might not be made today.

Portability was not a design issue, but several standards invoked on the project, mostly to promote maintainability, actually provided a reasonable basis for a portable system:

- Very few extensions of FORTRAN 77 were used;
- The only calls to IBM-specific functions are to open files and to the system clock;
- No machine specific data base management system;
- A core standard graphics package is used;
- The overlays are not embedded in the code.

Today, the software can be run under IBM MVS, VM/CMS, and MicroVAX on upward under VMS.

5. LESSONS LEARNED

The ARM project started in the Summer of 1983 resulting in a Phase 0 prototype installed in the Summer of 1984; a Phase I prototype in the Fall of 1984; and an operational system in the Fall of 1985. It is currently in a maintenance and modification mode. During these phases, there have been several hard-learned lessons that point out that no shortcuts can be taken in the development of an operational system. The main lessons were:

- Flexibility breeds user confusion and introduces a source of errors;
- Automation has an impact on operational procedures;
- Demonstration models only solve one aspect of the design problem;

- End user involvement is critical from the beginning of the project to foster total acceptance of and efficient transition to the use of the end product;
- Frequent user training is required in light of frequent user (military) turnover and program modifications;
- Acceptance testing should be conducted within the total operational environment--not as a stand-alone program;
- Redundancy in documentation is a configuration management nightmare;
- Documentation standards do not guarantee a document that leads the user through the system step-by-step.

The last five of the above problems are common to any system and need no further explanation. The first three have major implications for large automation efforts like ARM and deserve more elaboration. The need for flexibility in a large automated system cannot be denied. Flexibility in ARM is provided through parameterization. There are several hundred tuning parameters in the system that are used to ensure route quality. Some of them describe the planning heuristics, some the planning constraints, some the vehicle performance capabilities, and still others refer to the options such as terrain masking. This places a heavy burden on accurate documentation plus an overwhelming task on the selected users to review the documentation and the chosen data settings to ensure sound values. There is the added problem of determining which levels of users should have permission to change which variables.

The following steps have been implemented in ARM to help solve these problems:

- optional range of value check on each numeric variable, e.g., $0 < Ps < 1.0$,
- value check on monotonic sequences,
- value checks on some interrelated variables,
- user access levels for each variable,
- user write-access levels for each file, and
- common on-screen or off-line data base update capability.

Yet, none of these tools, nor extensions of them, relieve the primary data base manager from having to review the data base for consistency. What is needed for ARM and systems of its kind are front end systems that will review the data base and explain to the user the consequences of his composite data base. Such a system should be directed at individual modules and at the system as a whole.

Operational procedures are invariably altered by the introduction of a new tool that automates a portion of the procedures. The tool automates a process, but it in turn requires support (e.g., data base preparation, program monitoring, computer graphics replacing paper maps and drawing, additional emphasis on other processes). Hopefully, this does not take more time than the original process that has been automated! This change in procedures needs to be well thought out and new roles assigned to the staff who will use the tool well in advance of its introduction.

The final points are directed at demonstration models. A demonstration model may show a solution to be feasible but in fact the solution may not be practical when all the real world constraints are applied. In ARM's case, the enormous data bases and core space limitations were unknown when the model was built and were not considered in the model. Many of the algorithms used had to be modified to support these constraints. Thus, it is not necessarily true that the algorithms used for a prototype can or should be applied to a system requiring an operational data base many orders of magnitude larger than the test case. For other prototypes, the lesson is that these problems should be determined before the hardware is selected. In the ARM case, that was not an option. Fortunately, the ARM software was ultimately ported to a much more powerful IBM computer than the original target machine. This allowed the intermediate data base to be generated in about one day. This equates to a saving of months of human effort for each planning cycle.

A further problem with demonstration models for brand-new systems is that they may only solve one part of the problem. Figure 5 shows the components that contributed to the operational version of ARM. Only the portions of the problem indicated in gray were addressed in the demonstration model. This is typical of software resulting from research and development efforts. The prevailing thought is that the operational system should not cost very much since the problem has already been solved. The fact is that these other areas are equally important and become a driving cost factor.

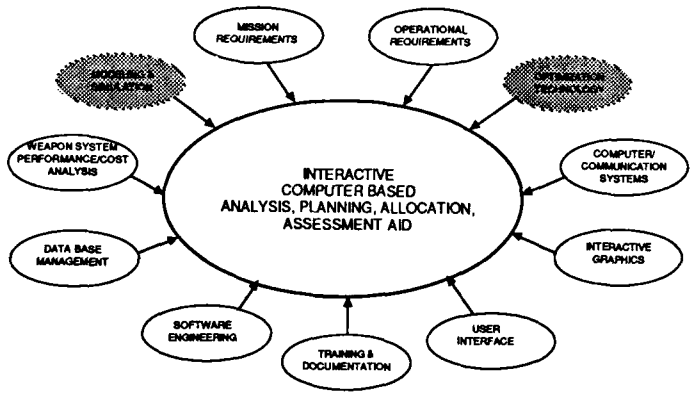

Figure 5 Successful Application of Optimization Requires More Than an Understanding of the Algorithms

6. CONCLUSION

ARM is a powerful and extremely flexible tool to support ALCM planning. The ARM project has shown that, with ingenuity, it is possible to automate a large problem. ARM has effectively automated the generation of candidate ALCM missions freeing the mission planners from tedious and time-consuming problems to plan more efficiently and concentrate on more difficult problems.

ACKNOWLEDGEMENTS

I would like to thank everyone on the ARM team from SCT, JSTPS, SAC, and DARPA who believed in the project and worked so hard to make ARM the success that it is today. Steve Rainbolt, Milt Grossberg, Jack Murphy, Kemp Deicke, John Russell, Sandy Henninger, Carol Tycko, John Mordeson, Tim Croll, Rich Vargus, Ellen Rubin, and Tina Youngs from SCT and Maj. Dale Richter, Maj. Clif Banner, Lt.Col. Dave Enos, and Maj. Fred Guice from SAC-JSTPS deserve special recognition for their roles in developing, guiding, and maintaining the ARM system.

REFERENCES

1. "User's Manual, 1986 Automatic Routing Module (ARM) Maintenance and Enhancement Software," B008, Systems Control Technology Inc., Palo Alto, California, September, 1986.
2. "Data Base Specification, 1986 Automatic Routing Module (ARM) Maintenance and Enhancement Software," B021, Systems Control Technology Inc., Palo Alto, California, September, 1986.
3. "System/Subsystem Specification, Program Specification, and Maintenance Manual, 1986 Automatic Routing Module (ARM) Maintenance and Enhancement Software," B016, Systems Control Technology Inc., Palo Alto, California, September, 1986.
4. "Software Interface Control Document, Volume I, 1986 Automatic Routing Module (ARM) Maintenance and Enhancement," B012, Systems Control Technology Inc., Palo Alto, California, September, 1986.