UAV Swarm Mission Planning Development Using Evolutionary Algorithms and Parallel Simulation - Part II
SCI-195

Gary B. Lamont
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Wright-Patterson AFB, Dayton, OH 45433
{Gary.Lamont@afit.edu}

Abstract

The purpose of this paper is to discuss the design and implementation of comprehensive mission planning systems for swarms of autonomous aerial vehicles (UAV). Such a system could integrate several problem domains including path planning, vehicle routing, and swarm behavior as based upon a hierarchical architecture. The example developed system consists of a parallel multi-objective evolutionary algorithm-based terrain-following parallel path planner, a multi-objective evolutionary algorithm (MOEA) for the UAV swarm router, and a parallel simulation. Generic objectives include minimizing cost, time, and risk generally associated with a three dimensional vehicle routing problem (VRP). The concept of the Swarm Routing Problem (SRP) as a new combinatorics problem for use in modeling UAV swarm routing is presented as a variant of the Vehicle Routing Problem with Time Windows (VRPTW). Various multi-objective VRPTW routing benchmarks result in very good Pareto-based performance with the MOEA which is also reflected in the results of the new SRP benchmarks. The culmination of this effort is the development of an extensible developmental path planning model integrated with swarm routing behavior and tested with a parallel UAV simulation. Discussions of this system's capabilities are presented along with recommendations for generic development of UAV swarm mission planning.

1 Introduction

Path planning is the process of designing a sequence of states through which an object must move in order to travel from an initial state to a goal state. Path planning optimization is a process that proscribes a particular plan for reaching a goal state from an initial state at a minimal cost. A path planning algorithm is a sequence of steps taken to calculate a path plan given knowledge of the path environment and a set of conditions or constraints that must be adhered to. Many successful path planning algorithms have been developed over many years [1,2,3,9,11,17,25,28]. These algorithms vary in their effectiveness and efficiency based primarily on the specific formulation of the path planning problem and the number of variables and constraints required. Based upon this foundation in part, it is desired to develop three dimensional (3D) autonomous aerial vehicles (UAV) mission plans including path planning, vehicle routing, and swarm behavior. The outline of the paper is: background, approach and objectives, mission planning, high level and low level design, implementations, experimental testing, results and analysis.

2 Background

An underlying element of UAV path planning is the *Vehicle Routing Problem* (VRP) which is defined as the task of assigning a set of vehicles, each with a limited range and capacity, to a set of locations or targets that must be visited [27] with cost and risk objectives. The VRP has been shown to be an NP-complete problem. Such problem classes do not lend themselves to deterministic problem solving methods because the runtime of these approaches grows exponentially with the problem size. Stochastic methods have been used to provide "good" solutions to the VRP in reasonable time [21,27]. These stochastic methods achieve their results by generating feasible solutions and then improving these results through successive refinements using heuristics.

The UAV *routing problem* consists of a set of targets L, a set of UAVs V, the set of traveling costs Q, the set of routes G, a distance function δ, a capacity function γ, and a demand function α. The formal definition is [21]:

$$\text{Given } L : \forall l_i \alpha(l_i) \geq 0, i > 0;\ \alpha(l_0) = 0 \text{ and } V: \quad \forall v_i\ \gamma(v_i) = k,\ k > 0$$
$$\text{compute: } Q : q_{ij} = \delta(l_i, l_j) \text{ and } G : g_k = \{l_0 \cup L \times L \cup l_0\}$$
$$\text{subject to: } \sum_{l \in G} \alpha(l) = \sum_{l \in L} \alpha(l) \text{ and } \cup l \in g = L \text{ and } \cap g \in G = l_0$$
$$\text{minimize: } \sum_{k=1}^{|Q|} Q_k$$

This model is addressed in the particular application which is a swarm of heterogeneous UAVs routed for reconnaissance or to deliver munitions to a set of targets in a selected terrain. A mathematical model for the VRP with time windows (VRPTW) and the associated swarm routing problem mathematical model are quite similar with additional constraints [37].

Representing cost and risk as fixed objectives is adequate for UAV routing problems in which distances between targets are large enough to ignore the added path lengths resulting from having to make series of turns in order to change heading from one location to another. However, when the target layout is such that the distances between the targets are as near as several turn radii of a UAV, then the cost of traveling between any two targets must consider the heading at which the UAVs arrived at the initial location and the heading they must assume to vector themselves towards the next target. Moreover, the relationship of the UAV swarm elements must be explicitly controlled. Taking this into account, algorithms that solve the VRP should calculate the cost of every assignment from scratch in order to accurately represent the cost associated with that assignment.

In this research, a UAV swarm path planning algorithm is developed that calculates the optimal route from a start node to an end node, through a mid point. This path through a triplet of locations can then be concatenated with other triplets to quickly and accurately calculate the actual cost of a vehicle assignment. This information can be tabularized and input to programs such as an evolutionary algorithm for solving the VRP. For example, the Genetic Vehicle Router (GVR) [21] where "good" assignments can be made but the costs associated with these assignments are more representative of the required physical route or path. The goal is not merely to calculate the true cost of a particular assignment made by the GVR but to influence the GVR to make better assignments using the more complete cost information and thus providing proper UAV turn corridors. Swarm behavior is of course an integral element of the generic UAV mission planning system in order to generate acceptable individual UAV altitude and attitude positions and velocities.

3 Approach and Objectives

When problems require minimization of multiple competing, cost elements, a trade-off is established between the set of competing requirements. In these instances, *multi-objective evolutionary algorithms* (MOEAs) can provide a decision maker with a variety of candidate solutions, each representing a level of optimization of one parameter with respect to another [4]. In this research, *a MOEA is developed for path planning where the objectives are cost, encompassing distance traveled and the amount of climbing a vehicle does, and risk resulting from flying through areas of threat*. The solution set contains a selection of routes such that each route has the lowest cost associated with a particular level of risk and vice versa.

Terrain Following (TF) is a mode of flight in which an aircraft maintains a fixed altitude above ground level (AGL) and flies low (on the order of a few hundred feet) through an area of interest. Naturally, this type of flying involves a great deal of climbing and descending, a costly operation. The TF concept is to remain hidden from enemy air defenses. The technique to hide within rugged terrain is known as terrain masking. Terrain Masking (TM) algorithms determine a route of flight in which an aircraft can move toward a target or location of interest while remaining masked from enemy air defense radar by the surrounding terrain. Often routes calculated by TM algorithms have significant climbing and descending costs associated with them. The process of picking the best-masked routes with the least possible cost in terms of climbing and overall distance traveled is known as Terrain Following Optimization (TFO).

Thus, the research goal is to develop mission planning capabilities for UAV swarms including VR, TF, and swarm behavior. In this effort there are *four main objectives:* **1**. Develop a multi-objective evolutionary algorithm for efficient path planning **2**. Develop a multi-objective router, **3** Develop a parallel system that computes individual route segments for input to a GVR algorithm and **4** incorporate swarm behavior throughout a parallel simulation.

The first objective concerns the development of a robust path planning algorithm for terrain following UAV missions. Since all routes have both a cost and a risk associated with them, path planning can naturally be expressed as a multi-objective minimization problem. Most often, decreasing the cost of the path, i.e. the path length and the amount of climbing required to navigate the terrain, results in increasing the risk associated with enemy air defenses. Likewise, a path generated to avoid intersection with all enemy air defense radar systems results in increased path cost. Single objective problem formulations for path planning often use constraints such as obstacle and threat avoidance and then calculate the least-cost path available that adheres to all constraints [22]. Other single objective problem formulations treat constraints as components of the solutions fitness [28]. Problems defined in this way have weights assigned to each objective and the resulting fitness is an aggregation of component scores. The common disadvantage of these approaches is twofold. First, a risk free path may not exist or its cost may exceed the UAV capabilities. Second, paths containing an acceptable level of risk may have a substantially lower cost than a completely risk adverse path if one exists. A multi-objective approach provides a choice of routes with cost proportional to their

level of risk. This empowers the decision maker to choose the acceptable level of risk and obtain the least-cost path associated with that choice.

The second objective focuses on the development on an effective router for directing the each sub swarm to the requested individual waypoints and leading to the specific targets. Numerous individuals have studied this problem and define a foundation for solving the swarm routing problem using this model [26,33,34]

The third objective using parallel path planning computation provides efficiency. Our associated Genetic Vehicle Routing algorithm [21,26] uses an evolutionary approach to find an optimal assignment of vehicles to targets for combat or reconnaissance missions. The algorithm uses as its set of inputs, the cost associated with traveling between any two target locations. This cost reflects only the Euclidian distance between the targets. In order to include the cost incurred by turning from one location and preceding to another, which increases the path length, the actual cost of traveling between two locations must include the direction from which the UAV swarm approached the first target and the direction the swarm departs the second target in route to a subsequent target. The generation of optimal route triplets scales as $\boldsymbol{O}(n^3)$ compared to the $\boldsymbol{O}(n^2)$ cost of optimizing pair-wise links. This limits scalability but is less costly than the exponential alternative of enumerating and calculating all possible permutations of complete route assignments. To offset some of the cost of enumerating triplets, the path planning algorithm is parallelized, solving multiple triplets concurrently. The output data from the path planner is then given as input to the GVR algorithm which has been modified to use this new data in its evaluation function. The result is an optimal assignment of UAVs to targets based on the true costs of completing the routes. Testing on this component focuses on the efficiency and scalability of the parallelization of the path planner and its ability to answer queries from the vehicle router.

Regarding the fourth objective for behavior evaluation, our swarm simulation model [5,10] represents a swarm of autonomous air vehicles with a set of three behaviors. The first swarm behavior is the tendency to remain together. The second behavior is a tendency to maintain a safe distance from one another. The third behavior is for the swarm members to align themselves together toward a particular direction. The swarm simulation is extended in this research to include a routing capability that guides the swarm along a route generated by the path planner and the GVR optimizer while still adhering to the three required swarm behaviors.

4 Mission Planning and Routing

Mission Planning for swarms of autonomous unmanned aerial vehicles requires an efficient assignment of vehicles or sub-swarms to targets, a set of efficient, feasible paths for vehicles to follow, a set of swarm behaviors that allow the swarm members to reach their targets while maintaining their collective swarm properties, and a detailed simulation of the mission to ensure objectives are met. This section considers historical approaches to solving these individual problems as well as a discussion of ways to unify these problem domains into a comprehensive problem statement.

Path planning: UAV path planning is a subset of a broader set of general path planning problems. All path planning problems and the algorithms used to solve them consist of some initial condition, objective, and a set of actions that completely connect the initial condition to the objective. However, there are many ways to specify a path planning problem. The method selected is often linked to the algorithm used to solve the problem.

Two broad categories of path planning problems and approaches dominate the research. The first category defines the problem in what is known as a *configuration space*. Problem formulations of this type involve determining the set of desired actions (torques, rotations, and other forces) needed to move a system from an initial state to a goal state. The second category of problem formulations, *trajectory spaces*, involves generating a set of feasible trajectories to move a vehicle from an initial location to a goal location.

In this research, paths are specified in line segments with restrictions on the degree of turn to ensure the path is navigable. Further, the concept of terrain masking which was loosely developed by Mittal [13] is extended with a complete terrain masking algorithm. The algorithm determines the maximum altitude (AGL) of an aircraft at a particular point such that at or below this altitude it is out of sight of a known threat - *intervisibility*. In addition to remaining out of sight of known threats, the terrain masking algorithm seeks to minimize the vehicle's exposure to unknown threats. This principle is known as *hidability*. It calculates the number of nearby points from which a vehicle is visible at a given altitude over a given point (see Figure 1).

Autonomous vehicles architectures: *Abstract autonomous vehicles architectures* for mission planning have been proposed by Reynolds [18,19] based upon a hierarchical game model, Gat [8] based upon a hierarchical control model and Price [15,16] based upon a finite automata self-organization model. Rysdyk [30] defines a trajectory following guidance architecture. Generally, desired complex goal-oriented behaviors are defined at the top of hierarchies and are produced by aggregations of lower level behaviors generally reflecting implicit or explicit state definitions.

Reynolds "game" hierarchal framework is: *Action Selection* (strategy, goals, planning), *Steering* (path determination), *Locomotion (*animation, articulation, control). Gat's three layer robot hierarchy is: *Deliberator* (goals, planning), *Sequencer* (plan execution), *Controller* (reactive feedback control, primitive behavior). Price's formal agent hierarchy is: *System state* (combined plans, environmental effectors), *UAV Agent state* (archetypes, behavior determination, path), *Update local state* (reactive action,

communication). Rysdyk's model is *world states, local states, vehicle states*.

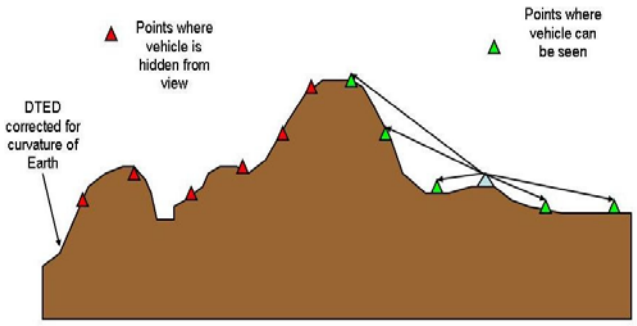


Figure 1 Principles of Hidability

Probing the details of these suggested frameworks, one would note that they are similar as regarding plans, behaviors, and implementations. Differences exist as to behavior specifics at each level, explicit interfaces between levels and use of associated formal notations. Reynolds for example developed a complex model for 3-D autonomous animation that was implemented for video games such as Sony's Play Station. Gat's architecture was developed for individual robot movement resulting in the ATLANTIS system. Price's model was used to develop a UAV swarm simulation with extensive environment interaction.

The physical model to represent a vehicle or physical agent is usually based on a point mass model consisting of a mass, position, velocity, maximum force, maximum speed, and an orientation with possibility of turn radii and moment of inertia.. The orientation can be given as a set of N-basis vectors and is therefore suitable for both ground and air vehicles. With the point mass vehicle model, the behaviors associated with the hierarchy act directly on its vectors. The low-level control signals which generate the primitive behaviors are communicated from the desired plan behavior. Behaviors under these hierarchies can include: *seek, flee, arrival, pursuit, offset pursuit, path following, obstacle avoidance, and containment*. Seek is the pursuit of a static target. It acts to steer toward a particular position. Flee steers the agent so that its velocity is radically aligned away from a fixed location. Pursuit is like seek but the added factor that the target is moving. This behavior requires not only knowledge of the target's velocity vector, but also the capability to predict the targets future velocity. Evasion is the opposite of pursuit i.e. the character is steered away from the predicted location of the moving target. Offset pursuit steers a path to come within and maintain a fixed distance from a moving target. Arrival is the same as seek when there is a significant distance between the vehicle and the target. However, arrival slows the vehicle down as it approaches. This behavior ends with the vehicle at a zero forward velocity and a position coincident with the target.

This view of behavior hierarchy addresses many of the requirements for a UAV swarm in order to be able to follow feasible paths to targets. The behavior set is rich and requires a complex set of individual members to execute. As to the level of autonomous self-organized UAVs, feasible paths can be generated by path planning module offline and assigned to swarm members or agents thus relieving them of burdensome computational requirements. At the strategic level of planning, the assignment of sub-swarms to target sets can also be performed offline allowing decision makers, rather than swarm agents themselves, to better guide the behaviors of the swarm to meet the goals. Within each of the suggested frameworks, such architectural variations can be selected. This then is the complex computational framework used in our UAV mission planning and routing system [23].

Evaluating our UAV routing performance is done on the AFIT UAV Swarm Simulator, a Parallel Discrete Event Simulation (PDES). Based originally on Reynolds' Distributed Behavior Model for flocking, the simulator was developed by Kadrovach [10] based impart on *Reynolds' Distributed Behavior Model* [18]. Corner [5,6,20] ported the model from a single-processor Windows platform to a parallel Linux-based Beowulf cluster. Slear [23] extended the model and integrated the mission planning generic framework into the current computational environment.

5 High Level Designs

The *high level system design* consists of three principal components: a parallel path planner, a vehicle router, and a simulation and visualization engine. The development of a comprehensive UAV mission planning system consists minimally of an efficient assignment of resources to targets, an effective means to create vehicle trajectories that minimizes risk to the resources and mission cost, and a behavior model that produces swarm behavior without degrading the other capabilities.

5.2 Parallel Path Planner

Two generic objectives are required: create an efficient and effective path planner using a MOEA, and create a flexible parallelization of the algorithm to allow for rapid generation of multiple paths for use in solving higher level optimization problems such as the *capacitated vehicle routing problem* (CVRP) [27].

The specific path planning problem model for UAVs consists of the following:

Given a discrete operational space of size n x m units

superimposed over a terrain grid $G \in (n-1) \times (m-1)$ *with*

Location set L, where $l_i \in n \times n \; \forall l \in L$

subject to: $\forall p_o \ldots p_n \in P, \; \Delta\theta\,(p_i, p_{i+1}) \leq 45^{\circ}$

where θ *is the inbound heading at* p_i

determine the least cost path P^* from all $l_i \in L$ to all $l_j \in L$

The restriction $\Delta\theta \leq 45^{\circ}$, ensures that the path remains flyable by the UAV. Based on the grid spacing of 750 meters, the UAV can safely navigate a 45-degree turn. This turn restriction can easily be modified to suit other vehicle types.

The term "cost" is a composite of individual objectives or measures of merit of a mission. In this research, five such measures of merit are defined (path, climb, terrain, detection, kill cost).

The *path* is the sum of the Euclidian distances of the route segments. *Climb* is the amount of climbing a vehicle must do in the course of flying a route in order to avoid terrain. The *Terrain* is the cost of exposure to unknown threats or the vulnerability associated with being "out in the open." *Detection* is the cost associated with being exposed to enemy detection – a function of both distance and time. *Kill cost* is the cost associated with being within the lethal range of an enemy air defense weapon – a function of range, time and the lethality of the weapon. While the problem domain of the generalized path planner has no restriction on the size of the target set *L*, the target set is limited to three targets or locations per instance, {Po, Pm, Pf}, to maintain compatibility with the problem domain of the CVRP which is solved by the router.

When a problem has five different cost functions (multi-objective), it can be solved as an aggregate function that attempts to simultaneously minimize all parameters, or it can be solved as a multi-objective problem where the output consists of a set of non-dominated solutions along the Pareto front. An end user can select one of these solutions provided they are capable of deciding the appropriate level of trade-off between two competing objectives. An output consisting of a five-dimensional Pareto front however, would likely overwhelm the decision maker by providing more questions than answers. Fortunately, the measures of merit can be grouped logically into two categories: those that describe the cost of the path in terms of time and fuel consumption, (*path* and *climb*), and those that measure the risk of a given path (*terrain*, *detect*, and *kill*). Equations 1 and 2 define the grouping of the five problem objectives into two competing categories.

$$\Phi_{cost} = \alpha\Phi_{path} + \beta\Phi_{climb} \quad (1)$$

$$\Phi_{risk} = \delta\Phi_{detect} + \lambda\Phi_{kill} + \omega\Phi_{terrain} \quad (2)$$

where $\{\alpha, \beta, \delta, \lambda, \omega\}$ are weighting factors associated with the relative importance of each parameter. The individual cost functions are:

Φ_{path}: The Euclidian distance between each point is summed over the length of the route.

$$\Phi_{path} = \sum_{i=0}^{f} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (3)$$

Φ_{climb}: The sum of positive changes in elevation from each point to the next point;

$$\Phi_{climb} = \sum_{i=0}^{f} \Delta z(p_i, p_{i+1})\delta \quad (4)$$

where δ is 1 if $Z_{pi+1} > Z_{pi}$ and δ is 0 otherwise.

Φ_{detect}: The total linear distance through which the UAV swarm flies into the effective detection ring of radar.

Φ_{kill}: The same formulation required for the detection cost function is applied to the kill cost function. The distinction between the two is the effective kill radius of an air defense system is generally smaller than the detect radius.

$\Phi_{terrain}$: While many threats are known a priori, others are not. Therefore, the UAV swarm should remain out of sight as much as possible. The terrain metric measures the number of points in the grid from which a vehicle at a particular point can be seen. The overall terrain score is determined by summing the surrounding points from which the vehicle can be seen as it flies though each grid point along its path.

MOEA path planner and router: A *multi-objective genetic algorithm path planner* interfacing to the CVRP router consists of the following elements: a population of candidate solutions, a defined chromosome structure of each candidate, a set of evolutionary operators which operate on the members of the population, a pair of evaluation functions to measure fitness of the solutions, an archived set of non-dominated solutions, and a defined period of evolution. The High Level View of MOEA Path Planning Algorithm is:

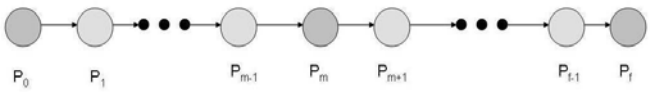
1: **procedure MOEA_Planner(*N* , *g*, $f_k(x)$)**
2: Initialize Population P of size *N*
3: Evaluate, Rank (by dominance), sort Population

4: Create archive population P_a from non-dominated members of P_i
5: **for** i in 1 to g **do**
6: Select for recombination (crossover)
7: **for** j in 2 to *N* **do**
8: Statistically select mutation operator Γ_k
9: Mutate member j
10: **end for**
11: evaluate Population
12: determine dominance rank within current population P_g
13: remove dominated members from P_a
15: add globally non-dominated members from P_g to P_a
16: **end for**
17: **end procedure**

The vector function $f_k(x)$ is the set of evaluation functions. In this algorithm, k=2 where $f_1(x)$ is the cumulative cost function and $f2(x)$ is the cumulative risk function. A population size of 50 individuals is selected along with an evolutionary period of 50 generations, g, based upon computational reasons. No heuristic was developed to terminate the evolutionary cycle once convergence of the solution was achieved. Further experimentation is needed to study the time saving benefits associated with early termination of the algorithm.

The chromosome structure is similar to that used in [67]. A chromosome of candidate solution consists of an ordered set of points (x_i,y_i) which define a path from the starting point (x_0,y_0) to a destination point (x_f,y_f) through a midpoint (x_m,y_m). Additional information contained at each point includes *elevation* (the MSL altitude of the point), *set clearance* (the AGL altitude of the point), and *heading* (the direction of travel from the present point to the next point).

Set clearance and altitude are used to calculate the amount of climb per descent needed to reach the next point as well as for terrain masking calculations. Heading is stored to ensure feasibility of the turns. The planner calculates the change of heading between points to ensure the turn rate is within the UAV's limits. The following diagram illustrates the chromosome structure of a



candidate solution.

During *initialization*, the population of candidates is created with each member containing the start, middle, and end points. An initial check is performed to ensure that the turn around the mid point is less than 45 degrees. If it is not, a modified convex hull algorithm is used to add additional points to the route such that no turn greater than 45 degrees remains. Once the route is repaired, a number of intermediate points are randomly added to the route. The number of points added is based on the distance between the three original points. During this process, the algorithm ensures that the change of heading between each point (excluding the starting point) is less than 45 degrees.

Once the population has been initialized [23], it is evaluated using the cost functions described. In a single objective EA, a program need only maintain the current population. In a MOEA, the complete set of non-dominated points must be maintained. An approximate Pareto front archive is maintained for this purpose. To find initial approximate Pareto front points, each member of the population is compared to every other member based on the member's F_1 score, Φ_{cost} and by its F_2 score, $\Phi risk$. The population is first sorted by Φ_{cost}. A candidate $\boldsymbol{R}_i$ is added to the approximate Pareto front if it meets the following criteria:

$$\forall p \in R, \neg\exists\, R_p \mid F_1(R_i) > F_1(R_p) \wedge F_2(R_i) \geq F_2(R_p) \qquad (5)$$

All non-dominated members of the population are then added to the Pareto Front Archive. The population is then sorted by rank. The rank of an individual R_i, reflects the number of individuals in the population that dominate *Ri*. All non-dominated members of the population are assigned a rank of zero. All members dominated by only a single solution are given a rank of one. Members dominated by two individuals are given a rank of two etc. Rank is the primary selection criterion used in the path planner. Dominance count is an alternative selection method. Dominance count is defined as the number of solutions in the population that a particular solution dominates.

A disadvantage of using dominance is that points along the ends of the front tend to evolve out of the populations while crowding occurs near the middle of the front. Rank is therefore preferable to raw dominance count because greater diversity is maintained in the population. Once the population has been evaluated and ranked, selection is performed. Like other MOEAs [3,4], the planner uses an *elitist selection* operator. The use of elitism is common in MOEAs because the elitism preserves non-dominated individuals. The top half of the rank-sorted population is selected for recombination. Pairing of individuals is done randomly. Once paired, two offspring are created. These offspring occupy the places of the members not selected.

Crossover is performed at the midpoint of the path. This ensures that the offspring remain feasible. During the one point crossover operation, the midpoints between two parents are exchanged. Since the underlying data structure is a linked list, the points beyond the midpoint are copied as well. The resulting offspring contain the points of one parent from the start of the path to the mid point, and the points of the second parent from the mid point to the end of the path.

Some other crossover operators considered include arithmetical, biased, multi-point, fuzzy forms, and uniform [4]. Because of the structure of the chromosome and the search landscape, the simple 1-point midpoint crossover provides the desired exploratory performance.

Once the crossover operator has been applied, the population then undergoes *mutation*. The path planner uses three distinct mutation operators which are applied with equal probability. The first mutation operator, M1 attempts to add a point between two existing points in the path. If the addition of the point results in an infeasible solution, then the repair operator is invoked to create additional navigation points. The sharper the turn created by the mutation, the more navigation points are needed to smooth the route. The repair algorithm generates a number of points proportionate to the change in heading caused by the infeasible point. For turns of just over 45 degrees, only two points are needed. For larger turns, as many as seven additional points need to be added. Therefore, when the mutation operation adds a point between two relatively nearby points, resulting in an unfeasible route, the path cannot be repaired and the operation is cancelled. Figure 2 illustrates this situation.

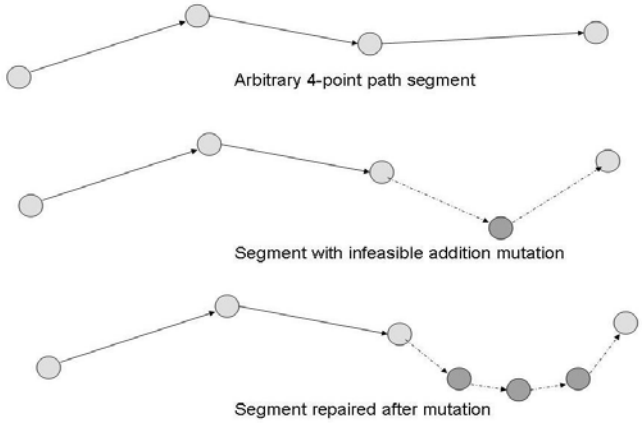


Figure 2 Mutate Add Operation on 4-point Path Segment.

The second mutation operator, M2, attempts to delete a point between two points in the path. Again, if the deletion results in an infeasible path, the repair operator is called to add points which result in a smooth trajectory. Deletion operations naturally increase the distance between points. Therefore, the repair operator is usually able to add the points necessary to achieve feasibility. Nonetheless, feasibility of the repair operation is still validated and if the path cannot be repaired, the operation is undone. It is important that balance is achieved between delete and addition operations. When too few deletions occur, the resulting path has too many points and is more difficult to evolve. When too few additions occur, the path tends to have very few points and the ability of the algorithm to minimize cost and risk is diminished. Because the deletion operation results in greater success, the addition operation is used with a slightly greater probability.

The last of the mutation operators, M3, selects an arbitrary point (not one of the original three) and attempts to alter its location by a bounded, random displacement. This operator does not change the number of points in the path by itself but additional points can be added when the alteration results in an infeasible path. When the bounds of the displacement are loose, the resulting path is more likely than not to be infeasible. Additionally, loosely-bounded displacement results in a greater number of points being added due to repair. On the other hand, if the bounds of the displacement are too tight then the operator becomes nothing more than a tool for local search. Possible additional mutation operators could enlarge the search space. For example, the use of exploratory mutation operators such as Xiao's Mutate 2 [28] that deletes multiple consecutive segments and replaces them with new ones could be considered. Rubio [30] uses a mutation operator in an EA along with market-protocol algorithms for path planning. Such techniques were not incorporated due to the additional complexity and concern as to generic utility in our approach.

Again, the swarm simulator must correctly route individual members to required targets by way of required waypoints. These way points are generated a priori as part of the path planner and are designed to minimize climbing, distance, and risk.

UAV Swarm Behavior: The problem of directing UAV swarm behavior can be expressed as the cumulative problem of directing individual UAV behavior. The following relations mathematically define the problem domain of the swarm model: Given a swarm member $\mathbf{v}_i$ and the following:

A terrain region (X,Y) with an elevation $Z = f(X,Y)$

A neighborhood vehicle set $\boldsymbol{V}$

A next waypoint w_{next} **< i, j, k >**

A current position s(t) = **< i, j, k >**

A set clearance C

Create a vector $\mathbf{v}(t+\Delta t)$ to guide $\mathbf{v}_i$ toward w_{next} subject to:

1. $z(t+\Delta t) > C + f(x\ t+\Delta t, y\ t+\Delta t)$ (6)
2. $|s(t+\Delta t)-w_{next}| < |s(t)-w_{next}|$
3. $\forall v \in V, v \neq v_i, |s(t) - b(v_i(t))| > |s(t+\Delta t) - b(v_i(t+\Delta t))|$ where condition 1 maintains the required set clearance, condition 2 moves the vehicle toward the next steering point, and condition 3 adjusts the separation between the member $\boldsymbol{v}_i$ and all neighbors in $\boldsymbol{V}$ toward the proper separation distance.

The behavior model consists of a set of rules to achieve path-following swarm behavior in a set of modes under which the rules are applied with various weighting factors, and a neighborhood of influence which defines which members affect the behavior of a given member. Each rule results in a unit vector addition operation applied to an individual. The sum of these vectors produces the member's trajectory.

Neighborhood - Just as with swarms of insects or flocks of birds, swarms of UAVs have limitations on information that can be obtained from other members of the swarm. These restrictions are generally based on the proximity of a member to other members of the swarm. In our model, we define the notion of neighborhood which is used to define the communication model as well as shape of the swarm formation. The swarm shape is a 3-D stack of diamond tessellations. Each plane or level in the stack is offset one half-step from the level directly above or below it.

The main parameter of the swarm formation is the separation *b,* representing the lateral distance between co-planar members and the distance of the co-planar neighbor directly in front and behind the member. Co-planar members 45 degrees front-left and front-right are at a distance of b divided by the square-root of 2.

Individual UAV swarms are not influenced by those behind them for two reasons. First, the lead members are first to climb in response to terrain and also reach their target and begin their turns before trailing members. Application of the cohesion rule would cause lead members to throttle back when climbing or turning to allow trailing members to catch up. Instead, catching up is achieved by trailing members applying the cohesion rule with respect to their distance from the leading vehicles. A second reason for this simplification is a reduction of the communication overhead. Restricting the neighborhood of a member to those members level with or in front of the UAV member, reduces the size of the neighborhood considerably. Table 1 defines the neighborhood of influence surrounding a given swarm member.

Table 1. Neighborhood of individual UAV influence

Plane	Distance	# Neighbors
Co-planer	b	3
Co-planar	$b/\sqrt{2}$	2
Plane Above	$b/\sqrt{2}$	3
Plane Below	$b/\sqrt{2}$	3
Two Levels Up	b	1
Two Levels Down	b	1
TOTAL		**13**

Rules. The behavior model consists of a set of three rules **R** = {r1, r2, r3} [45]. The application of these rules result from the interaction of individual swarm members with one another and with the terrain. As defined by Kadrovach [10] and implemented by Corner [3], each swarm member can only detect and be influenced by its neighbors. The first rule creates a vector that causes a vehicle to move toward its neighbor whenever the distance to that neighbor exceeds the threshold distance value. Recall that vehicles in the lead with respect to the next target are not influenced by the cohesion rule except by their coplanar members to the left and right.

Separation. Also from Reynolds, this rule adds a vector to the member moving it away from a neighbor when the distance to that neighbor decreases to below the threshold value. Leading vehicles have no members in front of them and are not directly influenced by those behind them. Therefore the separation rule applies only to their left and right co-planar neighbors and their neighbor's two planes directly above and below them. This rule replaces a more general alignment rule [3,10,18].

Modes. The simulation progresses under two primary modes: warp and synchronization. During warp mode, communication among swarm members is suspended. Individual members continue on their path at their current heading. When small changes in individual trajectories are needed to avoid terrain, the other members are not notified. An individual member simply adjusts its trajectory as needed. During synchronization mode, members determine their neighborhoods and adjust their trajectories according to rules 1 and 2. The simulation enters synchronization mode under two conditions: a) whenever a member alters its angular velocity by an amount greater than $\pi/8$ degrees, and b) at scheduled fixed time intervals. The later condition is required to prevent drift in the swarm which would occur if minor changes in trajectory are extrapolated over long periods of time. During warp mode, the members apply only rule 3 which accounts for climbing and descending. Under synchronization mode, the swarm applies rules 1 and 2 with a weight of 20% and it applies rule 3 with a weight of 30%. This weighting was established empirically for maintaining swarm characteristics while achieving the target seeking behavior.

Communication Model. The simulation is built on the SPEEDES time-warp framework [24]. Agent message traffic is restricted to neighbors and to the central simulation engine. This allows for true scalability of the UAV swarm model.

Since the entire swarm embarks on the mission from a single location, a swarm split must be performed as sub-swarms go out in search of their individual targets. In order to minimize maneuvering and communication required for a split operation, the swarm uses a train or sausage link model in its original formation. Upon reaching a designated split point, the leading section of the swarm becomes a sub-swarm and turns towards its next target. The remainder of the swarm turns toward its next target. The split is done along the length of the swarm like a section of railroad cars being removed from the track. This method has the advantages of maintaining the shape of the sub-swarm and reducing the swarm's temporal footprint. Once a swarm has split, there is no join operation defined. At the end of the mission, all swarms return to their embarkation point. Due to varying target assignments, the sub-swarms return home separately.

5.2 UAV Swarm Router

The purpose of this section is to discuss the implications of applying Multi-objective Evolutionary Algorithms (MOEAs) to the Vehicle Routing Problem with Time Windows (VRPTW). Specifically, as more constraints are applied to the VRP (as in the VRPTW) the solution space and Pareto front features change in such a way that multi-objective evolutionary approaches provide effective means of determining optimal solutions in a tractable time frame. Initial experimental results show the validity of this idea for the VRPTW. The concept of the UAV Swarm Routing Problem (SRP) as a new combinatorics problem for use in modeling UAV swarm routing is presented as a variant of the Vehicle Routing Problem with Time Windows (VRPTW). The genetic operators used are discussed in the context of how they contribute to finding better solutions. While some operators contribute random exploration aspects others contribute increasing value (decreasing path length) alterations to identified solutions..

5.2.1 Multi-Objective VRPTW Formulation Most often a VRPTW is optimized for path length. A second objective is the minimization of the number of vehicles used. In the VRPTW there are three objectives that can be optimized:

Total path length
Number of vehicles
Minimum waiting time

Waiting time is the amount of time a vehicle has to wait if it arrives at a customer too early. Minimizing this objective implies an efficient scheduling of all vehicles to all customers. One concludes that these two objectives are in contention in most problems, as efficient scheduling usually implies a lack of optimal path length (deploying two vehicles costs more but ensures minimum total waiting time). Optimizing effectively across both objectives allows for a more incremental search of the non-dominated front of solutions resulting in better optimal solutions. This is why multi-objective approaches to the VRPTW often lead to better results compared to biased single objective implementations [31] [32].

VRPTW Chromosome Structure Any chromosome solution used in a VRP must be able to specify how many vehicles are required and which cities must be visited in what order. The solution chromosome defines a genotype, which is a code corresponding to a phenotype which is the actual solution. In terms of total information the genotype does not need to contain redundant or implied information. For example, in the VRP it is implied that a route starts at the depot and ends there. Encoding this information in a chromosome would therefore be a waste of space. There are three ways to accomplish this, others could be formulated but these have been deemed effective through their repeated usage. A possible solution structure is a bit string where every bit corresponds to an edge in the solution (and every bit is either one or zero indicating whether it is or is not in the solution). This structure is very simple but grows large very quickly and the organization requirement of the VRP lend itself more toward real valued structures anyway. The second structure is a single array of real values, the order of which indicates the order of visitation. Each route is separated by zeros. This structure is more efficient but still requires the use of separators to indicate where a route begins and ends.. In [26] a structure for a VRP chromosome is defined that uses a similar idea as the array structure but attaches each route to a support structure, like that seen in Figure 3..The most beneficial aspect of this structure is that changes made to a given route do not require a shift to the entire array of values. In [48] this structure is proposed, and shown to be, an effective structure especially for the VRPTW. This structure is also used in previous research by Slear 23] and Russel [21].

The GVR structure offers many attributes that make it desirable as a chromosome structure. Its information content does not contain redundancies. Each route implies the existence of a departure and return to the depot even though it is not explicitly stated. This is made possible by the support structure that contains and separates each route. It is also desirable that infeasible solutions are not turned into feasible solutions by adding customers but instead only by rearranging and removing customers. The impact of this is that whenever a solution is checked for feasibility after the addition of a customer it can be safely discarded if infeasible, knowing the solution is a dead end.

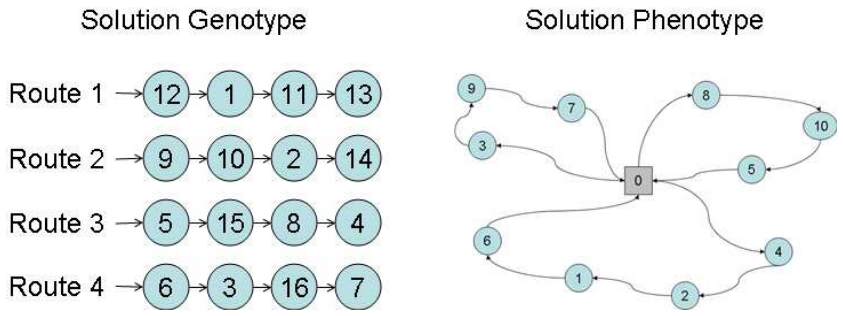


Figure 3. GVR chromosome structure for the VRP

Algorithmic Approach In order to define an evolutionary algorithm solution several algorithmic aspects must be identified. These are the chromosome structure of a solution, genetic operators, and selection methods. Other aspects can be identified however these are arguably the most relevant. In this section these three aspects are identified. A. Genetic Vehicle Representation. Tavares [5] defines a structure for a Vehicle Routing Problem (VRP) chromosome that uses a support structure containing each route. The support structure keeps each solution separated and organized. Changes then made to a given route do not require a shift of any values in the other routes, a beneficial aspect of this structure. This structures use shows it to be an affective chromosome, especially for the VRPTW.

Evolutionary Operators The specific operators are discussed in various papers [26,31,38]. All operators ensure that only feasible solutions are introduced into the population and each operator has a stochastic probability of occurrence.

1) Swap Mutation: In swap mutation two customers in a selected individual are swapped if doing so does not violate constraints.

2) Inversion Mutation: Select a sub-route and reverse the order of visitation within feasibility constraints.

3) Insertion Mutation: Move a random customer to a random location in the solution while ensuring feasibility. It is possible to create a new route containing this customer with probability 12V where V is the number of vehicles.

4) Best Cost Mutation: This heuristic based operator randomly chooses a route and optimizes its path length. To accomplish this, the customer closest to the depot is made the first customer then the remaining customers are rearranged in order of distance from each other. Customers that can not be feasibly added are moved to a new route.

Replacement strategy The multi-objective strategy incorporates a Pareto front replacement strategy where the best non-dominated solutions that are adequately spaced away from each other are assigned higher fitness values. Regarding the choice of MOEAs, the SPEA2 [35] and NSGA2 [36] algorithms are used in the experiments due in part to their universally.

5.2.2 Multi-Objective SRP Development

SRP Evolutionary Operator Development The SRP genetic operators are variants of the VRPTW operators altered to take into account the different structure of the SRP solutions. The difficultly in developing these operators is ensuring the validity of the child genotype. Since the chromosome contains location sensitive information across two dimensions, as opposed to the VRPTW chromosome which is only sensitive across a single route, making even slight changes can cause invalid solutions to be created.

SRP Chromosome Structure The chromosome structure used for the SRP is essentially the same as for the VRPTW. It consists of single route definitions arranged in a support structure. The only difference is the arrangement of data within the structure. Since each customer must be visited by more than one vehicle at a time the SRP structure must also reflect this possibility.

Random Crossover with Tightening. The crossover operation must be done with particular care as effective alterations to the solution are difficult to achieve. The basic idea of the crossover operation is the same as the VRPTW crossover operation, from two solutions a random route is selected from each. This route is then added to the other solution. The problem is, unlike the VRPTW crossover operation, subsections of a route can not be easily transferred between two solutions. In order to compensate for this the route to be crossed is added to the solution as an entirely new route. The solution then undergoes an operation called tightening. During this operation the solution is searched to determine what customers are over satisfied or visited at inappropriate times. The resultant solution contains the additional information of the crossover operation without the redundancy or errors the operation would otherwise result in.

Split Mutation. Split mutation randomly selects a route within the SRP solution and attempts to reduce the total length of that route be eliminating unnecessary target visitations. Each customer is satisfied with a certain number of UAVs at its location, however more can be present than are actually needed. This may cause a route to be longer than it needs to be since its divergence

to an unnecessary target takes longer than a direct route. The split mutation operation determines if this is occurring in a random route and attempts to remove the target from the vehicles fight plan. If this operation then results in an infeasible solution it is considered to have failed, and is not implemented.

Vertical Swap Mutation. The vertical swap operator swaps two different locations vertically in a given solution. This is in contrast to the VRPTW swap mutation in which the swapped targets can be anywhere. Columns within the SRP have a close approximation to time within the solution. It is not exact because distance information is not contained within the solution, and cities in the same column may not actually be visited at the same time. The swap operator randomly selects a column and two different targets within that column. These targets are then swapped and feasibility is checked. An infeasible solution is not used.

5.3 System level design goals and integration:

The system's data flow begins with creation of a target set, terrain field, threat lay-down, set clearance, and number of available swarm vehicles. The terrain masking algorithm is given the terrain elevation data, location and range of threats, and the set clearance or above ground altitude at which the vehicles fly. The threat lay-down is superimposed over the terrain grid and grid areas considered to be within the effective detection and kill ranges of the threat are identified. The algorithm then calculates the line of sight visibility of each grid space within the effective range. An individual grid space is eliminated from the effective range of the threat when a terrain barrier lies between the grid space and the threat such that a line drawn from the threat radar to the grid point intersects the terrain boundary thus obscuring the grid space from sight of the radar.

The set clearance of the UAV is added to the elevation of the grid space to account for the vehicles height above the ground. The updated threat range data is then stored for use by the path planner. Once the terrain has been preprocessed, the vehicle router optimizes the assignment of vehicles to targets. To accomplish this, the router needs to know the complete cost associated with a particular route. The router produces a set of candidate solutions and invokes the parallel path planner to provide complete, feasible paths for each route. The router's genetic algorithm finds the lowest cost vehicle assignment for the mission, and retrieves the complete set of waypoints for each vehicle or sub-swarm. This complete set of paths is then fed to the parallel swarm simulator which then simulates the mission and produces a visualization of the swarm flying its mission. Figure 4 illustrates the dataflow design of the integrated system.

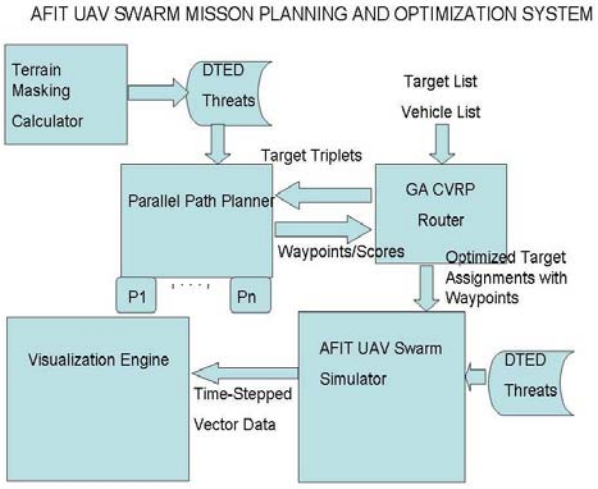


Figure 4 System Data Flow Design

The path planner produces a solution to the problem of minimizing the risk and cost associated with moving a vehicle from one location to another by way of an intermediate point. The input to the algorithm therefore, is a triple {*Pi, Pm, Pf*}. The output contains the set of waypoints between *Pi* and *Pm* and between *Pm* and *Pf*. This output forms a single segment of a solution to the larger vehicle routing problem which contains multiple targets and multiple vehicles. The router creates permutations of locations representing an ordered set of assignments to a set of vehicles. These permutations require sets of the triples described. The generation of these triplets is time consuming and the number of possible *3* triplets each grows at a rate of O(*n*) with the number of locations *n*. This growth rate is mitigated by three methods. First, the path planner is parallelized so that several paths can be generated at once. Next, the router uses a simple set of heuristics to request paths before they are needed. Finally, links which have already been calculated are cached for later reuse.

6 Low Level Designs and Implementation

6.1 **Path** Planner

The path planner was implemented using an object-oriented (OO) approach and is written in C++. The path planner has a naturally hierarchical structure. For example, a population consists of a set of paths, and a path consists of a set of points. This structure lends itself to object encapsulation. Methods are defined that act on objects at various levels of abstraction. Where the approach used differs from the traditional OO approach is in the area of information hiding. Typically an OO design defines strict controls on the access to an object's data members. Specific methods to access or alter an object are used to govern the range within data must be assigned and to control which objects are authorized to act on other objects. Details of the design can be found in [23] which has as the major goal of the system integration effort, modularity (population, path object, evaluation functions, …). The complexity of the various modules is polynomial.

6.2 EA SRP Router

The high level GA design necessitates many elements of preplanning before construction of a solution can take place. These decisions include coding language decisions, the use of GA libraries, and the level of generalization required. The level of generalization implies how dependent the software is on the problem and how difficult it is to transition the software to different problems. The coded solutions for the VRPTW, SRP, and path planner consist of a series of processing steps: read in problem data, maintain customer(target) information, construct individual solutions, apply the MOEA to those solutions, and return the results in a readable format. The implementation must be able to accomplish these steps and return results compatible with the simulation software. The software must adhere to object-oriented standards, be generalized enough to allow for the alteration of algorithm parameters, and be compatible with available hardware.

A basic structure exists within all genetic algorithm search techniques. This structure is defined by the transition of populations of solutions through a modification and selection process. Due to this structural concept it is advantageous to use a programming library or other software utility that has already been created with this basic structure in mind, hereafter refereed to as the infrastructure of the GA. There are a variety of infrastructure options available for evolutionary computation across many coding platforms. For this research the Open Beagle (OB) library was selected. Previous routing work used the GALib library [35], however for this research it was determined that a transition to a more contemporary library would be beneficial. The OB library contains very powerful and well constructed tools for the creation of evolutionary algorithms. It is written in C++ allowing for easier integration with existing simulation uses, all of which are written in C++, and the library allows the use of the vector data structure. The library is written in very strict object oriented protocol, meaning little work is required on the part of the user to get program specific details integrated into the overall program structure, assuming they are written to the same Object Oriented (OO) standard. More details concerning the implementation level design aspects of OB can be found in Appendix B and online [16]. The selection of this infrastructure drives the code level requirement of all the program components as well as the data structures available (C++ data structures).

7 Experimental Procedures

7.1 SRP Routing

This section contains information about the design of experiments perform. The objective of the experiment design phase is to develop testing methods for the problem model and solution design. The experiments divide into three sections which test the effectiveness of the routing software, the path planner, and the simulator. The purpose of these experiments is to validate the algorithm design of the routing software by applying standard VRPTW benchmark problems and comparing the solutions to best known solutions. Modified versions of these test problems are also applied to the SRP routing software in order for the results to be comparable.

7.1.1 Experimental Design Objectives;

Test the proposed solution design for application to the VRPTW is valid across a spectrum of benchmark problems, and the solution design for application to the SRP produces valid results comparable to those obtained in the corresponding VRPTW benchmark. These objectives drive the experiment design such that a set of benchmarks are applied to the VRPTW and SRP solutions resulting in a set of valid solutions. These solutions then contain measurable metrics of total path length, total vehicle count, total wait time, and average path length (these metrics apply to both the VRPTW and SRP). In the case of the SRP the benchmarks are modified such that customer demand (targets) is an indication of vehicle count and not capacity demand, as in the VRPTW. Comparison of these metrics of performance allows for an intelligent comparison of the solution process to benchmark

problems. Accurate performance comparisons require the application of different design choices to the same problem, using the same settings when possible. To fulfill this requirement genetic algorithm settings are chosen and kept constant across the spectrum of algorithm choices. These settings are determined through empirical experiments deemed to best represent the performance capability of the different genetic operators. Population size and generation limit are chosen within the desire to limit program run time. Three different algorithm designs, each with two options for selection strategies, are used in the experimental procedures. These three designs are NSGA2, SPEA2, and a biased elitism algorithm. The biased elitism algorithm uses no strategy to rank solutions instead using an elitist ordering procedure that is biased toward path length. The top number of individuals, equal to the population size are selected from the population after genetic alteration. Each of these designs is then paired with either a random or tournament selection process. Recall that selection refers to how solutions are selected for genetic mutation. Tournament selection means some number of random individuals is selected from the population, with replacement, and ranked (biased by path length) with the top rank selected for alteration. The SRP experiments employ only the use of the tournament selection method as random selection was deemed more harmful to the SRP solution process from the fact that the genetic operators employ no local search techniques.

VRPTW and SRP Experiments - The most commonly used benchmarks for the VRPTW are the Solomon problems developed in 1987 [38 They exist in three different varieties; a random distribution of customers (R), clustered sets of customers (C), and hybrid (RC). Each of these three problems comes in dimensions of twenty five and fifty customers. In order to examine the effectiveness of the software as well as the impact of the multi-objective design, two problems from each type are tested, listed in Table 6.1. The use of this variety of problems illustrates the impact of problem type on the solution design as well as solution performance in different instances. The number designation of each problem constitutes the time windows that exist for that problem. Problems that begin with a one, such as R109, have small time windows, while R206 has much larger time windows.

	Random	Cluster	Hybrid
25 Targets	R206 R109	C103 C205	RC107 RC202
50 Targets	R206 R109	C103 C205	RC107 RC202

Table 6.1: Solomon test problem selections (Modified for SRP)

Each test problem contains a set of target coordinates, target time windows, and vehicle capacity. The Euclidean distance between targets is considered to be the edge cost. The same problem selections are applied to the SRP solution modified in the demand column to ensure that each problem contains a realistic UAV requirement. Algorithm effectiveness varies greatly as different parameters within the program are tuned. The settings for each algorithm type were determined from empirical analysis and literature review [31]. The operator percentage indicates the chance that operator is used on an individual during the alteration phase. The more effective operators are used more often while the random operators are used less. All the options for the algorithm used to solve the VRPTW problems are listed in Table 6.2, the option for the SRP algorithm are listed in Table 6.2.

Operator and Setting
Random Crossover .4%
Swap Mutation .25%
Inversion Mutation .25%
Insertion Mutation .1%
Best Route Cost Mutation .4%
SPEA2 Archive Size 80
Generation Limit 1000
Population Size 100
Parent/children ratio 2

Table 6.2: VRPTW GA Settings

Operator and Setting
Random Crossover .5%
Split Mutation .25%
Vertical Swap Mutation .5%
Generation Limit 5000
Population Size 100
Parent/children ration 2

Table 6.3: SRP GA Settings

The SRP software experiments use the NSGA2 and biased elitism algorithms. The reason for this is that results from the VRPTW reveal a consistent dominance of these two methods over SPEA2. Each algorithm/problem experiment is run thirty times in order to ensure reliable statistical analysis. Each replacement strategy uses a tournament selection method. The population size and operator application percentages are different from the VRPTW settings in order to counter the SRP's fragile structure. More simple operations are performed to take the place of a few intelligent operations. Experiments are run against a small subset of the problems applied to the VRPTW.

Benchmark problems are run for the VRPTW and SRP solver using either a biased single objective or NSGA implementation. The benchmark is run 30 times for 1000 generations each trial with a population of 300. Solomon's problem RC107 for 100

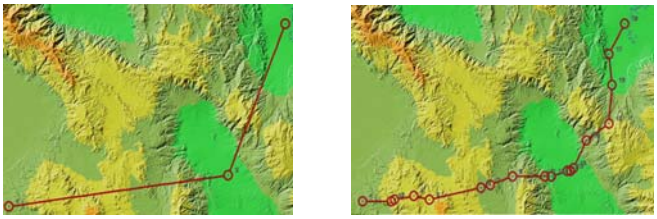
customers is used. All tests were run using a 2.1 GHz AMD Athlon 3000+, 1GB of memory, 128 KB of L1 cache, and 512 KB of L2 cache. Such experiments show the results obtained from the multiobjective formulation are superior to the single objective formulation, both in terms of the best path length attained and the optimization of multiple objectives. T he distribution of objective returns clearly favors the multi-objective (NSGA) approach over the biased single objective (Tournament) for not only path length but all objectives. Other empirical studies have shown this advantage becomes even more apparent as the dimension and complexity of the problem increases. Also note the much smaller distribution of values over the 30 trials for NSGA2 which indicates a much more consistent operation.

8 Swarm Parallel Simulation Experiments

The AFIT parallel swarm simulation uses SPEEDES which is an open-source parallel discrete simulation framework developed in C++. Its primary purpose is to allow users to develop small and large optimistic time-managed simulations [24]. Parallelization of the simulation allows for simultaneous processing of events. Optimistic processing of events enhances performance by allowing some events to be processed out of order. Out of order execution avoids delaying received events scheduled at a future time, while waiting on the receipt of all events from earlier times.

In the first experiment to minimize climbing, an artificial terrain field is created with a geometrically simple shape. The planner optimizes a route to the target by minimizing the climbing associated with the created path. Like all paths solved by the planner, this scenario consists of start, middle, and end locations. In between the straight-line path connecting the points are two large areas of high terrain which the planner must avoid. No threats are used in this experiment. Further, the weight associated with climb cost is maximized and the weight associated with distance is minimized to demonstrate the satisfaction of this single objective.

In illustrating tradeoffs between cost and risk experiment, a real-world route is planned over Nevada in the vicinity of Nellis Air Force Base. The path planner minimizes the cost of the route by minimizing distance, the amount of climbing associated with navigating the route, and the risk of the route Hideability, the degree to which vehicles remain out of site of potential unknown threats, is used as the optimization criterion. Figure 5 shows the three-point route for the planner to solve overlayed on a visualized *Digital Terrain Elevation Data* (DTED) field.



Three point route on DTED field Least cost and least risk route

Figure 5 3D Route planning in vicinity of Nellis AFB

This experiment compares the effectiveness of the path planner with a modified TFO algorithm. Three-element target packages are created along with a grid of real world terrain and a realistic threat lay down. The planner is run in single-objective mode and its solution is scored and recorded. The TFO is run and the output is passed into the evaluation function of the planner. The cost results are then compared.

There are two general uses for the path planner. As a stand alone unit, the path planner is multi-objective, i.e. it provides decision makers with a range of solutions to a particular problem instance. The second use is to generate link solutions to the larger capacitated vehicle routing problem (CVRP).

To *measure the efficiency of the parallel path planner,* the runtime of the serial version is compared with multiple-instance parallel runs of the algorithm. With this information, the scalability and speed-up of the algorithm is determined. The problem instance is suitable for this experiment because the route covers a wide range of the problem space,. Also, the repair function due sharp turn is tested, and the solution is overlaid on a varied, real-world terrain space where Terrain Following Missions are flown. The configuration of the test sets conducted on AFIT's Beowulf clusters consist of 1 to 16 processors and 1 to 1024 problems in intervals of powers of two. Each test is run 30 times for statistical analysis.

Modifications to the router only affected the data used by the routing algorithm. Naturally, scores from the path planner differ greatly from the static point-to-point scores originally used by the router. Experimentation in this area focus only on the ability of router to: successfully invoke the planner, make use of the planner's path scores, and complete its genetic routing algorithm.

In the previous version of the swarm model, a 2-D swarm was placed into a uniform terrain region with targets or points of interest. The model demonstrated limited capability to find targets while conforming to the swarming rules [23]. A suite of experiments is developed to test the effects of the additional model capabilities on the swarm model. Table 2 illustrates the behavior enhancements of as they relate to the previous model. Testing focuses on adherence to the swarm rules as defined and the scalability of the enhanced model. The three major behavior enhancements are tested independently and collectively.

Table 2 Swarm Model Behavior Enhancements

Model	Swarm Dimension	Attraction Rule	Repulsion Rule	Path Following	Terrain Following
Previous	2-D	Yes	Yes	No	No
New	2-D and 3-D	Yes	Yes	Yes	Yes

Specific actions taken by the vehicles to reach targets, often conflict with the swarming behavior rules. These experiments test the ability of the swarm to maintain its physical integrity while reaching all assigned targets in the route. Each experiment uses a common set of information to determine the adherence to the swarm rules.

Neighborhood: A neighborhood is calculated for each swarm member at each time step in the simulation output. Each neighbor has a required separation parameter based on its position relative to the central UAV as defined in the parameter file.

To separate consequential rules violations from minor ones, a threshold violation level is set at 20% of the separation parameter. Rules violations in this experiment are then determined by instances when a vehicle's separation from any of its neighbors differs from its required distance by ±20%. Note that various swarm splitting is required for sub-swarms to follow each CVRP route.

In order to observe the effect of the path-following behavior on the swarm's cohesiveness, metrics are required. From the simulation data, the average neighborhood size is calculated over time. Deterioration of neighborhood size is indicative of the swarm spreading out beyond its intended range.

For the first experiment, the simulation is executed over flat terrain with only a single vertical layer. This configuration allows for isolation of the effects of path following from other model enhancements. For each time t in the simulation, the average neighborhood size is calculated using:

$$Ave.N.Size\ (t) = \sum_{t=0}^{f} |n| \ / \#UAVs \quad at\ time\ t \qquad (7)$$

Another measure of compliance is the degree to which rules are violated. To measure the degree of violation, the absolute value of each UAVs violation in meters is calculated at various time steps in the simulation. Equation 7 quantifies the magnitude of rules violations for UAV i at time t:

$$\sum_{j=1}^{n} |vectdiff\ (i,j) - req.separation\ (i,j)| \ / \ n \qquad (8)$$

where *vectdiff(i,j)* is the separation vector between the *i* and *j* UAVs and *req.separation* is the position-dependent separation distance required by the model's rules.

Various experiments are executed to evaluate the impact of cost and risk minimization along with terrain following employing the indicated metrics. Swarm behavior such as synchronization, rule adherence, cohesiveness, sub-swarm shape with terrain following are analyzed over the 3D layered UAV model. Parallel scalability evaluation was addressed via a speedup factor with configurations consisted of 4, 8 and 16 processors simulating 40, 80, 160, 320, and 640 UAVs.

9 Results and Analysis

To test the planner's ability to minimize climbing, an artificial terrain field is created with a geometrically simple shape. In this case, the planner-generated route avoids the high terrain to eliminate climbing. The planner is run in multi-objective mode and the least cost and least risk solutions are captured and visualized. Figure 5 also shows the optimized route for cost and risk minimization.

The route in Figure 5 was scored according to the fitness functions. Its component scores are given in Table 3. Figure 6 shows a visualization of the lowest risk score. These two solutions represent the two extremes of the Pareto front. To compare the planner to the terrain following optimizer, this experiment analyzes the effectiveness of the path planner with modified TFO algorithm. This experiment was performed by running the problem instance Nellis Route 1 on the path planner and comparing the results with TFO's solution. It should be noted that TFO was not able to solve the problem directly. Due to algorithmic constraints of TFO's tree search, a maximum of 20nm are allowed between targets. As a result, intermediate points had to be inserted between the targets before the route could be optimized. An additional limitation of TFO is that it optimizes paths between targets but does not optimize connections between targets. Therefore, TFO does not allow more than 45 degrees of heading change between consecutive major waypoints. The parallel path planner has neither of these constraints. Figure 6 depicts the TFO solution to the problem instance Nellis Route 1.


Lowest risk route

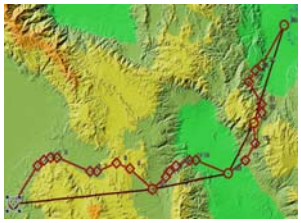

TFO route

Figure 6 Lowest risk and TFO 3D routes over Nellis AFB

Table 3 Fitness Component Scores - Nellis Route

Route	Path Length	Climb Cost	Hideability
Low Cost	94289	10522	14774
Low Risk	93857	12444	17904

Several inferences can be made from inspection of Figure 6. First, TFO's approach to minimizing climbing involves seeking the lowest point possible. Inspection of the path between shows that higher terrain was avoided whenever possible. This contrasts with the parallel path planner's approach which focused on minimizing the total amount of climbing. While TFO would avoid high terrain at any cost, the parallel path planner allows for high terrain so long as the cost of moving into the terrain is offset by reduced climbing and descending within the terrain. Table 4 compares the fitness evaluation of the TFO solution with the low cost and low risk Pareto front points of the parallel path planner.

Table 4 Nellis Route Evaluations

Path	Path Length(m)	Climb Cost	Hideability
Low Cost	94289	8655	15696
Low Risk	93887	10272	14671
TFO	110931	11460	21758

Table 4 shows that both solutions of the parallel path planner had lower risk routes with shorter path lengths. The planner's low cost route found a lower climb cost while the TFO found a lower climb cost than the low risk route. While the two programs have a different approach to minimizing climbing, it should be noted that the hideability algorithm and its input data are identical in both programs. Figure 6 also reveals a weakness in TFO's application of the restriction on heading change. Recall that consecutive target inputs in TFO must not result in a change in heading greater than 45 degrees. In the problem tested, as each segment was optimized independently, the resulting solution contains a heading change greater than 90 degrees. The MOEA resulting Pareto front for the same problem instance is given in Figure 7 providing multi-objective tradeoffs to the decision maker.

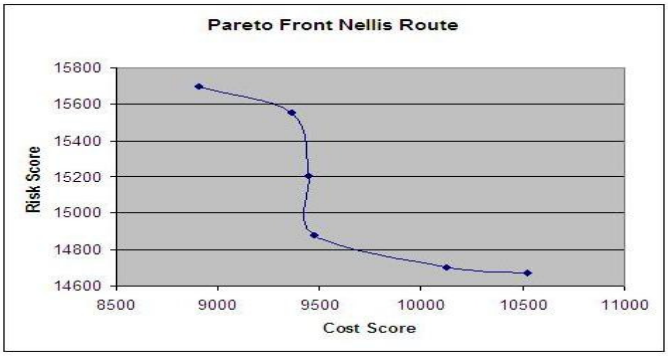


Figure 6 Risk vs. Cost Pareto Front

The efficiency of the parallel path planner experiments reveal near linear speed-up. This is due to the independence of the nodes, and low communications overhead. Runtimes varied from one job on one processor of 0.2 seconds to 123.3 seconds for 1024 jobs on one processor to 8.5 seconds for 1024 jobs on 16 processors. It is clear that the parallelization of the path planner results in near linear speedup with each increase in the number of processors. This result is not unexpected as the parallel decomposition strategy has very low overhead. It should be noted however, that the load balancing scheme and the use of

multiple non-blocking receives contributed to the speedup. In the absence of effective load balancing and non-blocking communication, the speedup would be reduced even with low-overhead parallel problem decomposition.

Evaluating the expanded and improved parallel swarm simulator was also a critical element in the development of the UAV mission planning system [23]. Most insight as to the performance of the UAV mission planner was achieved with the parallel simulation as well as feedback to improve planning and routing effectiveness [29].

10 Conclusions and Future Research

Multi-objective evolutionary algorithms are developed for efficient UAV swarm path planning and routing. Development of the new Swarm Routing Problem (SRP) model is shown to be an effective and solvable problem model for multiple UAV routing. Also, an efficient parallel computation system is developed that computes individual segments for use in the GVR routing algorithm. The parallel swarm simulator is improved by incorporating path-following capabilities with existing swarm behavior and measuring the effects of these capabilities on swarm characteristics. Additional efforts include exploring larger-sized physical areas for terrain and threat avoidance. Another promising technique is to increase the search space through the use of "migrant" population members. Originally developed for use in the Island model [26], migrant members are randomly initialized solutions added to the population at various epochs of the evolutionary cycle. Modifications to the path planner should allow either validation that time on target constraints can be met or that adjustments in the vehicle speed can be evolved along path segments. The addition of more population diversity would allow the planner to search different regions of the problem space. In particular,, a dynamic parameterized UAV vehicle second-order model tuned to create path feasibility would also be of practical importance. In addition, qualification of sensor data requirements would provide a realistic simulation that could lead to real-world swarm implementation and testing, The bio-inspired UAV swarm, agent model [39] should also be investigated for embedding.

Acknowledgements: This effort is a research element of the AFIT Advanced Navigation Technology Laboratory. The research is sponsored by the Information Directorate and the Sensors Directorate (Virtual Combat Laboratory), Air Force Research Laboratory (AFRL), WPAFB, Ohio. In addition the following graduate students supported the development of the material: Adam Pohl, James Slear, Charles Russell, Jim Corner, and A. Kadrovach as well as Ken Melendez, a staff scientist.

Bibliography

1. Brown, Darrin. *Routing Unmanned Aerial Vehicles While Considering General Restricted Operating Zones.* MS Thesis, AFIT/GOR/ENS/01M-04. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), WPAFB, Dayton, OH March 2001.

2. Carriker, W. "The Use of Simulated Annealing to Solve the Mobile Manipulator Path Planning Problem," *Proceedings of the 1990 IEEE International Conference on Robotics and Automation* (1):204-209 (1990). 9.

3. Castillo, Oscar and Trujillo, J. "Multiple Objective Optimization Genetic Algorithms for Path Planning in Autonomous Mobile Robots," *Int'l Journal of Computers, Systems, and Signals,* 6 (1):48-62 (2005). 14 November 2005

4. Coello, C. D. Veldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems,* Kluwer Academic Publishers, Norwell, MA 2002. 2nd Edition, 2007

5. Corner, J. and Lamont, G. "Parallel Simulation of UAV Swarm Scenarios," *Proc. Winter Simulation Conf.*, 355-363 (2004)

6. Corner, J. *Swarming Reconnaissance Using Unmanned Aerial Vehicles in a Parallel Discrete Event Simulation*. MS Thesis, AFIT/GCE/ENG/04-01. Graduate School of Engineering and Management, Air Force Institute of Technology, WPAFB, Dayton, OH 2003

7. ----SkyView. *Georgia Tech Research Institute*. 5 Jan 2005

8. Gat, E., "On Three-Layer Architectures," *Artificial Intelligence and Mobil Robots*, *Case Studies of Successful Robot Systems*, pp195-210, AAAI press, 1998

9. Jia, Dong. Vagners, J., "Parallel Evolutionary Algorithms for UAV Path Planning*." Proceedings of the AIAA 1st Intelligent Systems Conference*. (2004)

10. Kadrovach, A. *A Communication Modeling System for Swarm-based Sensor Networks.* PhD dissertation, Graduate School of Engineering and Management, Air Force Institute of Technology, WPAFB, OH (2003).

11. LaValle, S. *Planning Algorithms,* Chapter 5, pp 193-196 Cambridge University Press, MA. 2006

12. Lozano, P. "A Simple Motion-Planning Algorithm for General Robot Manipulators," *IEEE Transactions on Robotics and Automation*, 3(3), 224-238 (1987)

13. Mittal, S and Deb, K. "Three-Dimensional Offline Path Planning for UAVs Using Multiobjective Evolutionary Algorithms," Kanpur Genetic Algorithms Laboratory Report: 20040008, Indian Institute of Tech., Kanpur, India (2004).

14. Nikolos, I., Tsourveloudis, N.C., VAlavanis, K.P., "Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* 33 (6):898-912, 2003.

15. Price, I. *Evolving Self Organizing Behavior for Homogeneous and Heterogeneous Swarms of UAVs and UCAVs.* MS Thesis, AFIT/GCS/ENG/06M11. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), WPAFB, Dayton, OH March 2006.

16 Price, I. and G. B. Lamont, "GA Directed Self-Organized Search and Attack UAV Swarms," *Proc. Winter Simulation Conference,* 2006

17. Rathbun, D. "Evolutionary Approaches to Path Planning Through Uncertain Environments," *Proc. of AIAA UAV Conference*, Portsmouth, VA #2002-3455 (2002).

18. Reynolds, C. "Flocks, Herds, and Schools: A Distributed Behavior Model," *Computer Graphics*, 21(4):25-34 (1987).

19. Reynolds, C. "Steering Behaviors for Autonomous Characters," *Proceedings of The Game Developers Conference.* San Jose, California. 763-782. (1999).

20. Russell, M. "On Using SPEEDES as a Platform for UAV Swarm Simulation." *Proceedings of the 2005 Winter Simulation Conference* 1130-1137 (2005).

21. Russell, M., *A Genetic Algorithm for the UAV Routing Problem Integrated with a Parallel Swarm Simulation*. MS Thesis, AFIT/GCS/ENG/05M-16. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), WPAFB, Dayton, OH March 2005.

22. Sezer, E., *Mission Route Planning with Multiple Aircraft & Targets Using Parallel A* Algorithm*. MS Thesis. AFIT/GCS/ENG/01M-06. College of Engineering, Air Force Institute of Technology, WPAFB, Dayton, OH (2000)

23. Slear, J., *AFIT UAV Swarm Mission Planning and Simulation System,* MS Thesis, AFIT/GCE/ENG/06-08. Graduate School of Engineering and Management, Air Force Institute of Technology (AU), WPAFB, OH June 2006.

24. -----. *SPEEDES Users Guide*. Solana Beach CA: Metron, Inc., April 2003 (DN: 2024 v4)

25. Sugihara, K and Smith, J. "A Genetic Algorithm for 3-D Path Planning of a Mobile Robot." white paper, Department of Information and Computer Science, University of Hawaii, Manoa, Honolulu, 1999

26. Tavares, J. "GVR Delivers it on time," *Proc. of Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02),* 745-749. 2002.

27. Toth, Paolo and Vigo, D. *The Vehicle Routing Problem*, Philadelphia: SIAM (2002).

28. Xiao, J. and Michalewicz, Z. "Adaptive Evolutionary Planner/Navigator for Mobile Robots," *IEEE Transactions on Evolutionary Computation* 1 (1):18-28, April (1997)

29. Melendez, K., Slear J., and Lamont G., "Parallel UAV Swarm Simulation With Optimal Route Planning," *Proc. of Summer Simulation Conference*, 2006

30 Rubio, J.S., Vagners, J., Rysdyk, R., "Adaptive Path Planning for Autonomous UAV Oceanic Search Missions," *AIAA 1st Intelligent Systems Technical Conference*, 2004

31 B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Applied Intelligence*, vol. 24, no. 1, pp. 17–30, 2006.

32. B. Barn and M. Schaerer, "A multiobjective ant colony system for vehicle routing problem with time windows," in *Proceedings of the 21st IASTED International Conference*, Innsbruck, Austria, February 2003.

33 K. Tan, L. Lee, Q. Zhu, and K. Ou, "Heuristic methods for vehicle routing problem with time windows," *Artificial Intelligence in Engineering*, vol. 15, pp. 281–295, 2001

34 K. Q. Zhu, "A new genetic algorithm for vrptw," in *International Conference on Artificial Intelligence, Las Vegas, '00* .

35 E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

36 K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Paris, France: Springer. Lecture Notes in Computer Science No. 1917, 2000, pp. 849–858

37. A. Pohl, "Multi-Objective UAV Mission Planning Using Evolutionary Computation," M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB (Dayton), Ohio, March, 2008

38. A.. Coello Coello and G. B. Lamont, Eds., *Application of Multi Oobjective Evolutionary Algorithms*, ser. Advances in Natural Computation. World Scientific Publishing, 2004, vol. 1.

39. D. Nowak, "*Exploitation Of Self Organization In Uav Swarms For Optimization In Combat Environments,"* , M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB (Dayton), Ohio, March, 2008