A Genetic Algorithm for Unmanned Aerial Vehicle Routing

Matthew A. Russell and Gary B. Lamont
Dept of Electrical and Computer Engineering
Graduate School of Engineering & Management
Air Force Institute of Technology WPAFB (Dayton) OH, 45433-7765, USA
ptwobrussell@mac.com, gary.lamont@afit.edu

ABSTRACT

Genetic Algorithms (GAs) can efficiently produce high quality results for hard combinatorial real world problems such as the Vehicle Routing Problem (VRP). Genetic Vehicle Representation (GVR), a recent approach to solving instances of the VRP with a GA, produces competitive or superior results to the standard benchmark problems. This work extends GVR research by presenting a more precise mathematical model of GVR than in previous works and a thorough comparison of GVR to Path Based Representation approaches. A suite of metrics that measures GVR's efficiency and effectiveness provides an adequate characterization of the jagged search landscape. A new variation of a crossover operator is introduced. A previously unmentioned insight about the convergence rate of the search is also noted that is especially important to the application of *a priori* and dynamic routing for swarms of Unmanned Aerial Vehicles (UAVs). Results indicate that the search is robust, and it exponentially drives toward high quality solutions in relatively short time. Consequently, a GA with GVR encoding is capable of providing a state-of-the-art engine for a UAV routing system or related application.

Categories and Subject Descriptors: I.2.6 [Learning]: Miscellaneous, I.2.8 [Problem Solving, Control Methods, and Search]: Heuristic Methods

General Terms: Algorithms

Keywords: Vehicle Routing Problem, Genetic Algorithm, Genetic Vehicle Representation, Unmanned Aerial Vehicle

1. INTRODUCTION

The Vehicle Routing Problem (VRP) is one of the most common problems in practice and dates as early as organized civilization itself in the form of goods distribution, trading, and public transport [16]. Graph modeling formulations of this abstract problem definition is the base of many combinatoric problems that run in real-time such as Unmanned Aerial Vehicle (UAV) routing [1, 6, 12, 14, 18].

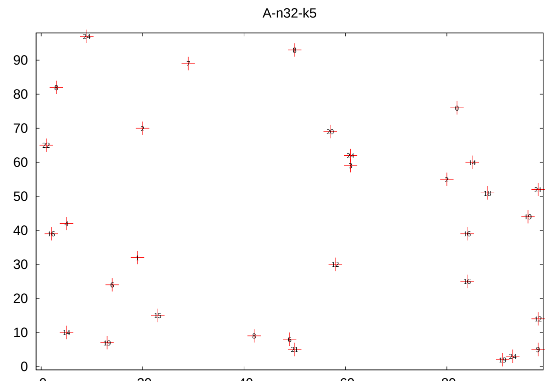
In addition to already being an NP-complete problem, the VRP is often subject to a plethora of constraints including heterogeneous vehicle types, delivery time windows, minimizing total cost, time travelled or the number of vehicles, and asymmetric routing [28]. Consequently, VRP combinatorics exceed that of other hard problems such as the traveling salesman problem (TSP) and the bin-packing problem (BPP). In fact, the largest solvable instances of the VRP are two orders of magnitude larger than those of the TSP [20]. Myriad combinatorics and highly practical applications demand finding high quality VRP solutions as quickly as possible. Genetic Algorithms (GAs) can efficiently and effectively fulfill this computational demand.

2. PROBLEM AND ALGORITHM

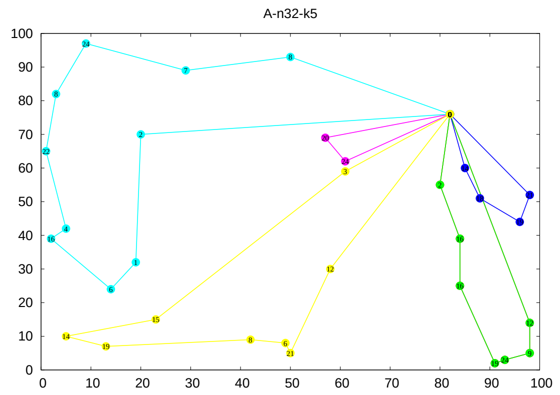
Vehicle Routing Problem The most basic VRP formulation is that of the Capacitated VRP (CVRP). The CVRP defines the problem in terms of a fleet of homogeneous vehicles with each member of the fleet having a fixed capacity that may not be exceeded. Tuples of the form $(location, demand)$ exist and must be serviced subject to the capacity constraints of the vehicles. The objective of the problem is to minimize the overall cost of supplying the locations via the delivery fleet; the total Euclidean distance traveled is typically the cost metric.

As applied to UAV routing, the CVRP defines a set of locations L each requiring a reconnaissance payload known *a priori* c_k, such that $\forall l_i \exists c_n$, where $n > 0$. A fleet of UAVs K having maximum capacity $c_{vehicle}$ is located at location l_{depot} and is available to fulfill the requested reconnaissance payloads according to a cost from the set $Q \ni q_{ij}$, which is incurred by traveling amongst any two locations l_i and l_j. Let G be a sequence obtained from $L \times_k L$ that starts and ends at l_{depot}. The objective of the CVRP is then to find a set of simple circuits given by $Z \ni min \sum_{g \in G} \sum_{q \in Q} q$ subject to the following additional constraints: No partial reconnaissance payloads are allowed, $\forall l_m \exists c_i \ni c_i = c_m$, all reconnaissance payloads must be fulfilled, $\forall l_i [\exists q_{ai}, q_{ib} \in Z]$, and the only intersection of the circuits is at l_{depot}, $\bigcap_{z_i \in Z} = l_{depot}$. Figure 1 illustrates a small CVRP benchmark problem of relatively low combinatorics[1] and its optimal solution. CVRP benchmark problems are well addressed in the literature [5] and are typically of the form *X-nYY-kZ*. X refers to the problem class, Y to the number of loca-

[1]Using the TSP's asymptotic complexity as a gross underestimate, this instance's combinatorics are $\Omega(n^n)$. Using this metric, there exist at least 1.46×10^{48} possibilities to consider in the solution space.



(a) Each coordinate on the graph is a delivery location and has an associated delivery demand. Deliveries begin and end at a central depot.

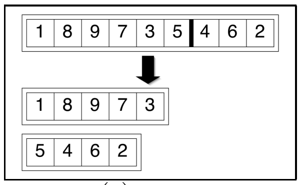


(b) The optimal solution minimizes the total Euclidean distance travelled.

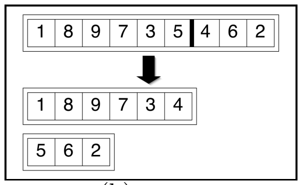
Figure 1: Benchmark problem A-n32-k5 and its optimal solution.

tions, and Z to the theoretical minimum number of vehicles required. Problems usually cluster, equally spread, or strategically place delivery locations. The average 'packedness' ratio for the theoretical minimum number of vehicles is also a benchmark design factor. Having much applicability to everyday commerce and transportation, the VRP has been approached from nearly every angle imaginable. Some specific approaches from the literature include Ant Colony Optimization [8, 13], Tabu Search [22], Evolutionary Strategies [11], parallel approaches [2, 17], Simulated Annealing [7], problem simplification [31], fuzzy logic [16], and branch-and-bound techniques [20]. GVR [15, 19, 24, 25], the focus of this discussion, is a recent approach with special emphasis on the data representation. An *efficient* data representation is the key to lowering overall computing times; it is especially important in GAs, which are stochastic as well as iterative in nature.

Path Based Representation A naive method for representing VRP solutions is with path-based representation (PBR). In PBR (PBR-1) a solution to the VRP is simply a single sequence of locations. In such a sequence, routes are obtained by walking the sequence from start to finish in order to group consecutive locations into groups. When adding the next consecutive location would violate a feasibility constraint (exceeding the total capacity of a vehicle), the current route is terminated and a new route begins with the next consecutive location. This particular decoding procedure takes $\Theta(n)$ time for n locations. Figure 2(a) illustrates PBR-1. The actual genotype is depicted as a single route with a dark line to designate the earliest point the need for a repair can be detected, if locations are processed by walking through the array. The dark arrow represents the repair process that leads to the feasible solution. A variant of PBR-1 is shown in Figure 2(b) as PBR-2. When possible, this operator more tightly packs the repaired solution than PBR-1. Instead of simply segmenting the current route, any possible location that could be added to the current solution from the remaining locations is added. This particular procedure in its worst case could potentially take $O(n^2)$ time and would involve each vehicle only being capable of serving a single location. Thus, all possible attempts to add an additional location are in vain. The worse case is not expected in any real-world problem, because an optimal solution would be trivial: use a single vehicle for each delivery location. Still, a worst case time complexity calculation gives an upper bound on the decoding time and is trivial to prove with mathematical induction. The best case for PBR-2 is given when the sequence of locations optimally packs the vehicles. In this case, the decoding sequence takes only $O(n)$ time.



(a) PBR-1



(b) PBR-2

Figure 2: PBR solution decodings. Even locations have a demand of 1 unit, while odd have a demand of 2 units. Vehicle capacity is 10 units.

Genetic Vehicle Representation GVR is the work of Tavares et. al [15, 19, 24, 25] and provides a solid base on which to build an effective UAV routing algorithm. The strength of GVR is its data representation; its novel aspects include its crossover and mutator operators. A more precise presentation of these operators than has been previously presented in the literature is paramount to developing better metrics for GVR and to controlling its balance between exploration and exploitation.

GVR differs from Path Based Representation (PBR) in that it *can* explicitly provide the number of routes and locations in them for a given solution without a significant decoding burden. For any given solution, there exist a set of routes where each route is a sequence of locations. In previous works of GVR [15, 19, 24, 25], each route may or may not meet feasibility criteria. In the case that a route fails to meet feasibility criteria, it is repaired as in a Baldwinian evolutionary model [4]; a copy of the solution is repaired and evaluated, but the actual building blocks of the solution in the GA population remain unaltered. A Lamarckian model [4], on the other hand would have repaired the original solution, evaluated it, and left the newly repaired solution in the GA population. Given these two models, the time complexity to decode a solution varies between $\Theta(n)$ for the case in which solutions are explicitly repaired and exactly n

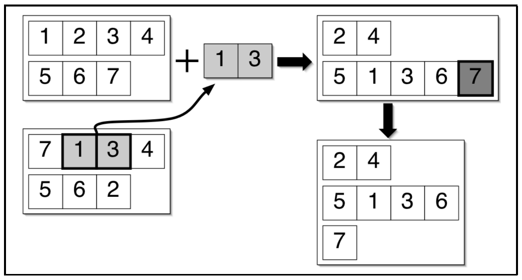


Figure 3: The crossover in GVR The final repair step occurs with $0 \leq p \leq 1$.

locations are visited with each decoding, and $O(n^2)$ for the worse case in which no repairs take place and a mechanism such as PBR decodes the solution.

Crossover Operator The GVR crossover operator is conceptually simple. Two individuals, C_1 and C_2 are chosen for selection. A subroute, r from individual C_1 is inserted into a copy of individual C_2 in a way that the distance between the insertion location and the first location in r_1 is minimized. Mathematically, $\exists k \forall l dist(k, r_1) < dist(l, r_1) \wedge k \neq r_1 \wedge k, r_1 \in L$. In general, the crossover operator requires $\Theta(n)$ time for n locations in a solution since all possible solutions must be examined. For a given subroute r and donor S of size n, there exist n possible crossovers.

Clever techniques such as table lookups or hashes may decrease this time to a constant, but even then, it may still take $O(n)$ time to traverse the routes and insert r. Depending upon the particular implementation, an additional preprocessing time of $O(n)$ may be necessary in order to remove the duplicate locations from the surrogate such that once r is inserted, the solution is again feasible. Figure 3 illustrates the GVR crossover. The step illustrating the repair operation occurs with some probability $0 \leq p \leq 1$. Note that the crossover operator can never create additional routes, because there always exists a location in an existing route that has a minimal distance to r_1. The crossover operator, however, can eliminate an existing route. This happens only when the subroute r already exists in the donor as its own route.

Swap Mutation In swap mutation, two locations l_1 and l_2 in solution S are chosen at random and swapped with one another. These two locations may or may not exist in the same route, and there exists the possibility that both random locations may be the same (in which case no change occurs as the result of the mutation). Thus, for a solution containing n locations, there are n possible swap mutations that may take place. Figure 4(a) shows the swap mutator.

Notable properties of the swap mutator are that the number of locations in the affected routes does not change: $\forall R_{pre-swap} \forall R_{post-swap} \in S \; [|R_{pre-swap}| = |R_{post-swap}|]$. The length of the routes and total demand for the route, however, almost certainly changes. Additionally, swapping may result in a feasibility violation by causing a route to exceed the maximum capacity of a vehicle. The potential disruption for the swap mutator is comparatively low because at most two locations are affected. Consequently, even if frequent swaps occur in a model with low repair probability, the overall characteristics of the solution remain relatively similar for the general case.

Inversion Mutation For inversion mutation, a subroute

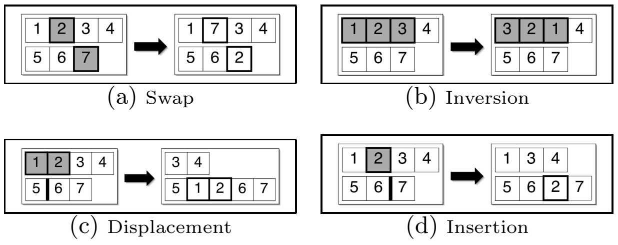


Figure 4: GVR mutation operators

from a solution $r \subseteq S$ is chosen and inverted. For a solution containing n locations, there are $\Sigma_{i=1}^{n} \binom{n}{i}$ possible inversions including the trivial ones in which a subroute of size 1 is inverted, and no actual mutation occurs.

Notable properties of the inversion mutator are that the number of locations and total demand of the affected route does not change, $\forall R_{pre-inv} \forall R_{post-inv} \in S, |R_{pre-inv}| = |R_{post-inv}| \wedge dem(R_{pre-inv}) = dem(R_{post-inv})$. Thus, the feasibility criteria is not affected as a result of the mutation. The only change that may occur is the total distance of the route. Inversion mutation, however, can be very disruptive, especially in models with very low repair probability. The reason for this disruption is because the inversion mutator can change a subroute of a solution in which the subroute is actually all of or parts of two separate routes. Figure 4(b) gives the inversion mutator.

Displacement Mutation The displacement mutator selects a subroute from a solution $r \subseteq S$ and relocates it to another place, which may or may not be in the same route. In fact, it is identical to the crossover operator except that it randomly inserts the subroute r instead of inserting it in such a way that the total distance after the operation is minimized. This random insertion point may result in the creation of a new route with some probability; Recent approaches use the probability $\frac{1}{2V}$, where V is the number of vehicles in the current solution [15, 19, 24, 25]. Thus, the higher the number of routes in a solution, the lower the probability that an additional route is inserted.

Like the crossover operator, for a solution containing n locations there exist $\Sigma_{i=1}^{n+1} i$ total possible displacement mutations including the trivial cases in which no change occurs. For a given subroute r of cardinality k, there exist $(n+2)-k$ possible displacements in the solution. The possible disruption associated with this mutator is expected to be relatively high since possibly long subroutes may be displaced. Figure 4(c) shows the displacement mutator.

Insertion Mutation Insertion mutation is a special case of displacement mutation in which the subroute size of displacement is of size one. Thus, for a solution containing n locations, there are $(n + 1)$ possible insertion mutations including the trivial case in which no change occurs. Potential disruption from insertion mutation is expected to be very low since at most one route and one location is affected. This holds true even in models with low repair probability in which very long infeasible routes may exist in a solution. Figure 4(d) illustrates the insertion mutator.

Generating Feasible Solutions There exist three mainstream approaches for handling feasibility criteria with a GA. One approach is to generate an initial population of feasible individuals, and design operators that guarantee feasibility after their operation. The overwhelming disadvantage of this approach is that it is not always a straight-forward process and can be difficult or impossible to implement efficiently for the crossover operator, especially with integer-represented solutions reflected in the TSP or VRP. Given this difficulty, repair and penalty functions become the two notable alternatives.

Repair Functions A repair operator transforms an infeasible solution into a feasible one. This approach has the advantage over the previous possibility in that the crossover operator can be more simply and efficiently implemented. As a result, more creativity and thought can be placed on its design, and it can be more easily maintained or updated. A decision that must be made when designing a repair operator, however, is whether to repair explicitly as in a Lamarckian model, or implicitly as in a Baldwinian model. In either case, a repair may occur with some probability $1 \leq p \leq 0$.

Penalty Functions Another possibility is not to pose any explicit feasibility constraints on a solution; rather, use a penalty function that guides the search towards feasible solutions by rating infeasible solutions with lower fitness values[2]. A penalty function should produce solutions to the problem that are feasible, and has the potential benefit over the repair function in that it does not explicitly manipulate the building blocks to guide the search. Still, effective penalty functions are usually not trivial to derive and often require problem specific information.

A common type of penalty function is one that applies a static penalty to solutions that violate feasibility in any way. One formulation of a static penalty function is given by Equation 1 [4]:

$$f_p(x) = f(x) + \Sigma_{i=1}^{m} C_i \delta_i \qquad \begin{cases} \delta_i = 1 & \text{if } i \text{ violated} \\ \delta i = 0 & \text{otherwise} \end{cases} \tag{1}$$

A disadvantage of this approach, however, is that it assumes that the penalty is a function of the number of constraints violated, which may not be the case. Satisfying constraints for some optimization problems requires solving additional NP-Complete problems. Additionally, suitable coefficients must be specified for this approach to be effective depending upon the relative severity of the constraints to one another. An additional concept to investigate when using a static penalty function like Equation 1 is whether or not penalizing for constraint violations can actually guide the search towards feasible areas. In problems with a semi-chaotic landscape, a static penalty function may be of little use. Nonetheless, its success or failure is still an additional insight into the search landscape.

Another approach that is generally superior to Equation 1 penalizes according to the 'cost-to-completion' [9, 21]. Equation 2 illustrates such a penalty function [4]:

$$f_p(x) = f(x) + \Sigma_{i=1}^{m} C_i d_i^{\kappa} \qquad \begin{cases} d_i = \delta_i g_i(x) & \text{for } 1 \leq i \leq q \\ d_i = |h_i(x)| & \text{for } q < i \leq m \end{cases} \tag{2}$$

In Equation 2, κ is a user-defined exponent, and d_i is the distance metric of constraint i applied to solution x. Constraints $1...q$ are inequality constraints and are activated when the constraint is violated. Constraints $(q+1)...m$ are equality constraints that activate the penalty if there is any distance between the solution and constraint values [4]. Thus, the inequality constraints guide the search toward the feasible solutions, and the equality constraints guide the search toward local or global optima. Although existing literature for static penalty functions is quite mature, defining parameters for penalty functions remains far from trivial and very much problem dependent [26]. Additionally, the notion of 'distance' between feasible and infeasible solutions is hard to precisely define for hard optimization problems, especially ones with a viperous landscape.

[2]This approach assumes an exterior penalty function.

Adaptive penalty functions can be viewed as a particular implementation of Equation 2. Adaptive penalty functions increase penalty values in monotonically nondecreasing manner proportional to the search time, which may be the number of generations. Equation 3 illustrates an implementation of Equation 2, where $s_i(t)$ is the monotonically nondecreasing function that satisfies the definition of an adaptive penalty function as presented. An explicit example of $s_i(t)$ might be an expression such as $s_i(t) = C_i t$ or some variation thereof.

$$f_p(x,t) = f(x) + \Sigma_{i=1}^{m} s_i(t) d_i^{\kappa} \tag{3}$$

3. EXPERIMENTATION

Experiments in this suite are designed to provide insight into GVR's efficiency and effectiveness. These experiments collectively form a suite and the outcome of each experiment successively determines a point of interest for the following experiment. This is to say that preliminary experiments are somewhat broadly designed, and each successive experiment becomes more narrowly focused based on previous results. For implementation, the GA is designed with GAlib [29] and runs on a 32-bit 2.2GHz AMD Opteron Processor with 4GB of memory. Data from each experiment is collected to qualitatively fulfill each experiment's objective using basic first order statistics.

Experiment 1 *To qualitatively determine the extent that a static penalty function can guide the search process and lead to effective solutions using GVR.*
The objective function is crucial to successful search with a GA. An important design decision impacting a GA's effectiveness is how it handles infeasible solutions generated from the recombination operators. GVR very often generates infeasible solutions from the crossover operator; the infeasible solutions could be either repaired or penalized in order to guide the search. Previous approaches for GVR used repair functions for guiding the search [19, 23–25].The intention of this experiment is to determine to what extent a static penalty function can guide the search process. Feasible CVRP solutions pose two basic constraints: 1) No vehicle capacities can be exceeded, and 2) At least the minimum number of routes possible must appear in a solution.

The problem definition provides vehicle capacities, and a lower limit on the number of possible routes is given by dividing the total demand of all delivery locations by the vehicle capacity. Equation 4 penalizes both criteria; for the former condition, it penalizes using a 'cost-to-completion' metric, which is shown to be more effective than penalizing based only on the number of violated constraints [4].

$$f_p = \Sigma_{i=1}^{R} f_{cap} \cdot f_{veh} \qquad \begin{cases} f_{cap} = P_c l_i \delta_c \\ f_{veh} = P_v \delta_v \\ \{\delta_c, \delta_v = 0\} \quad \text{iff i satisfied.} \end{cases} \tag{4}$$

This experiment runs all possible ratios of penalties, p_{cap} and $p_{veh} \in \{0, 1, 2, 4, 8, 16, 32, 64, 128, 256\}$ on benchmark problem A-n32-k5 [5]. This problem, as previously mentioned, is of relatively low combinatorics and is optimally solved in well under 10,000 generations by previous GVR research using a repair-based approach [24, 25].

Experiment 2 *To measure the effectiveness of the repair operator in guiding the search process.*
The GA is run on a diverse test suite of common CVRP benchmarks using a repair operator that segments infeasible solutions at the point of violation in the route.[3] The benchmark problems include A-n32-k5, A-n54-k7, A-n69-k9, B-n63-k10, E-n76-k8, and M-n200-k17. Results are compared to the best published in the literature and to results from other algorithmic techniques.

For all problems except A-n32-k5 and M-n200-k17, population sizes of 200 with 50,000 generations of evolution are tested. For problem A-n32-k5 only 10,000 generations are tested, because this is an ‘easy’ benchmark problem and most published results use this metric. For problem M-n200-k17, a population of size 400 and 100,000 generations are used because the combinatorics of the problem are much larger than the other benchmarks under test. Furthermore, a matrix of runs is conducted for all possible combinations of $p_{repair} \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and for all possible mutation combinations from $p_{mutation} \in \{0.05, 0.10, 0.15, 0.20\}$ except for two cases. Because problem M-n200-k17 is very large and takes a long time period to execute, the mutation combinations are reduced to $p_{mutation} \in \{0.10, 0.15, 0.20\}$. For this experiment, tournament selection with tournament size $q = 5$ and crossover $p_{cross} = 0.75$ is used. The tournament size is designed to apply very little selection pressure during the search, while the crossover rate is intended to exploit the effects of recombination. Given various solutions to these benchmark problems, visualization is accomplished via graphing the maximum, average, and minimum fitness values over time.

Experiment 3 *To determine how dependent effective problem solutions depend on tuning mutation parameters.*
Effective mutators are crucial to gaining insight into a GA’s ability to explore. Given the best results from problem M-n200-k17 in Experiment 2, four matrices of runs with fixed crossover rate and variable mutation rates are constructed. Graphical display of the data is accomplished by holding two of the mutators at constant values while varying the others over a fixed range for 0.02 increments. This grid size should be fine grained enough to collect data indicative of the landscape. The combinations of rates being varied include: displacement and insertion, swap and inversion, inversion and displacement, and swap and displacement.

Experiment 4 *To determine if there exists a statistical significance between two different repair operators; one operator segments routes at the point of capacity violation, while the other packs the existing route as tightly as possible.*
Given the best results from problem M-n200-k17 in Experiment 2, this experiment determines whether there is a significant difference in solution quality for two different repair operators. One operator repairs by packing the infeasible routes encountered during objective function evaluation as tightly as possible. Another repair operator, the one used in Experiment 2, simply segments infeasible solutions at the point of the violation and creates another route with the infeasible portion. For each of the two operators, a series of 11 runs with different random number seeds is conducted.

[3]Experiment 4 compares this operator to one that attempts to pack routes as tightly as possible by fitting any possible locations into the remaining space.

Table 1: Summary of benchmark results

Problem	p_{rep}	p_{swp}	p_{inv}	p_{dis}	p_{ins}	fitn.	best	% diff
A-n32-k5	**	**	**	**	**	784	784	0
A-n54-k7	**	**	**	**	**	1167	1167	0
A-n69-k9	0.5	0.15	0.15	0.15	0.1	1164	1159	0.4
B-n63-k10	0.5	0.05	0.15	0.1	0.1	1507	1496	0.7
E-n76-k8	0	0.15	0.1	0.1	0.05	741	735	0.8
M-n200-k17	0.5	0.1	0.15	0.1	0.1	1348	1296	3.9

Experiment 5 *To determine how dependent the search process is on tournament size and if population reinitialization after a time of stagnation can improve search results.*
Selection pressure is proportional to the ratio of the tournament size q to the population size. Given graphical displays of maximum, mean, and minimum fitness values over time from Experiment 2, this experiment determines if doubling q and reinitializing the population each time the minimum fitness value in the population does not decrease by more than $\epsilon = 10$ units can improve solution quality. This approach is tried on the best and worst solutions for problem M-n200-k17 from Experiment 2.

Experiment 6 *To determine if high quality solutions generated from the GA can be further improved by a lambda exchange local search algorithm [26] as a post-processor.*
Given a high quality solution produced with the Genetic Algorithm, this experiment determines if the results can be further improved by means of a fast local search that systematically displaces all combinations of up to two locations. Results from this experiment provide additional insight into the proximity of ‘good’ solutions to either local or global minima.

The following section presents results from these experiments and the analysis needed in order to ascertain the impact of GVR encoding for UAV routing applications.

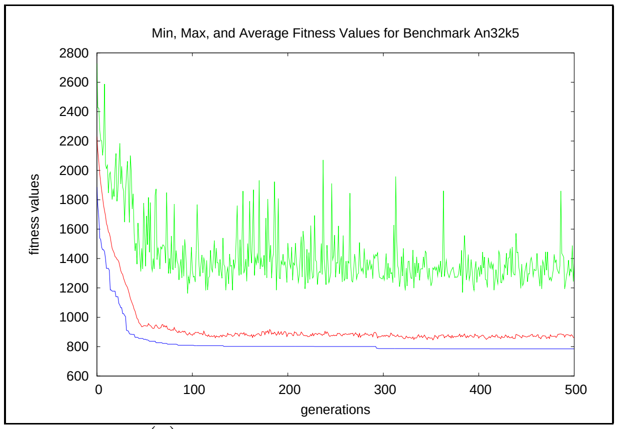
4. RESULTS AND ANALYSIS

Given results from the suite of experiments, analysis is tailored toward determining the efficiency and effectiveness for using the GA and GVR encoding for real-time UAV routing applications. Basic statistics are adequate to analyze the data collected.

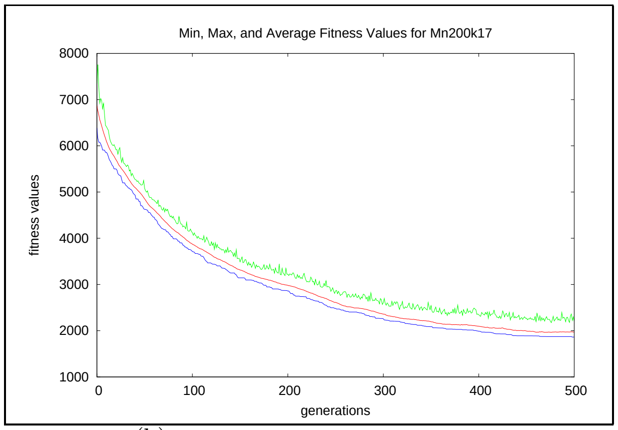
Experiment 1 For all ratios of penalty functions given by Equation 4, no solutions were feasible. Given that problem A-n32-k5 is considered an ‘easy’ benchmark problem, these results suggest at a minimum that Equation 4 is not an effective static penalty function, and more likely, that static penalty functions are not effective techniques for guiding the search process. Results from Experiment 2 produce results competitive with the best-published results on A-n32-k5 in under 300 generations. It appears that penalizing solutions to the problem is not effective in part because of the sheer combinatorics. All other experiments in this suite use repair functions to guide the search process. Future work in investigating penalty functions should inspect adaptive or dynamic approaches [4].

Experiment 2: The GA produces optimal or near optimal results for the CVRP benchmarks tested. All results are more than adequate for effective vehicle routing requiring limited time for *a priori* computation. Known optimums are quickly reached for all tested parameter combinations in problems A-n32-k5 and A-n54-k7. Near optimal and high quality solutions for all other parameter combinations are produced for all other problems.

The high quality results GVR produces are desirable, but it is important to determine how fast the search converges on these results if it is to provide the engine for a route planning system that demands near optimal solutions as quickly as possible. Mission planning does not necessarily entail an optimum solution; rather, solutions within a particular tolerance are often acceptable. Additionally, it might be noted that the exponential increase in solution quality suggests that this algorithmic technique is likely to provide competitive results for instances of the Dynamic VRP, in which delivery schedules might change while vehicles are en route. A graphical display of the search progress over time reveals



(a) Generations 1-500 of A-n32-k5



(b) Generations 1-500 of M-n200-k17

Figure 5: Solution quality exponentially increases

that a period of rapid improvement in solution quality is followed by long periods of stagnation for all benchmark problems. In the end, this type of convergence is almost ideal for fast *a priori* routing. This suggests that a local search or a similar technique could be used during this period of stagnation instead of continuing the global search with the GA, although the GA does continue to gradually improve results. It is also evident that GVR increases the number of good building blocks at an exponential rate, per the Schema Theorem[4] [30]. Gradually improving partial solutions in a problem like the VRP is more than reasonable; locations close together gradually link into routes, and routes that complement one another have a higher probability of survival and continued improvement. The results from Experiment 5 present the results from increasing the selection pressure and reinitializing the population during these periods. Results from Experiment 6 present the results from trying a local search as a post-processing routine.

[4]Acceptance of the Schema Theorem is not unanimous [27].

Table 2: Search times for selected benchmarks.

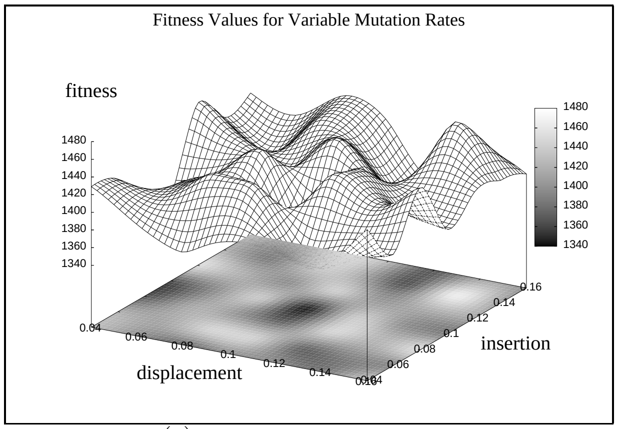
Benchmark	sec/gen	termination (gen)	routing time
A-n32-k5	0.00142	500	7.15 sec
A-n69-k9	0.0292	5000	2.43 min
B-n63-k10	0.0277	10000	4.61 min
E-n76-k8	0.0311	10000	5.18 min
M-n200-k17	0.158	15000	39.5 min

Figure 5 shows the max, min, and average fitness values of each generation for benchmark problems A-n32-k5 and Mn-200-k17. Problems A-n63-k10, A-n69-k9, B-n63-k10, and E-n76-k8 result in the same net effect: a period of rapid increase in solution quality is followed by a period of stagnation with very little improvements occurring. Only the first 500 generations are shown. Note that the elitism employed in the GA results in a monotonically non-increasing minimum fitness value over time. The fluctuations in the maximum and average fitness values illustrates that exploration is taking place, but the stagnation of minimum fitness values indicates that the search space is treacherous once solutions become nearly optimal. A fair description of the search space at this low level amounts to something like trying to sink a golf ball into a hole surrounded by potholes and ruts from 100 yards away using only a bent putter.

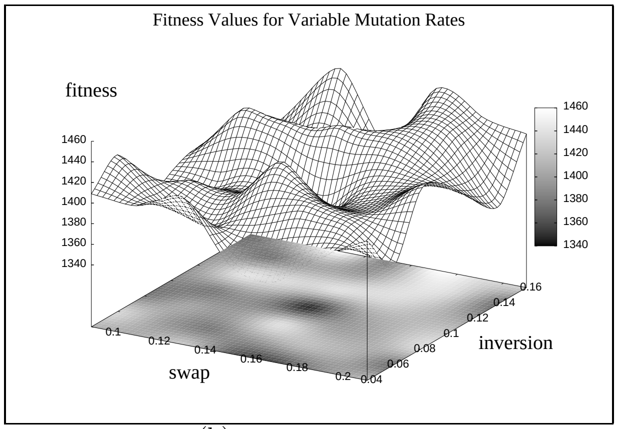
Table 2 hypothesizes a *a priori* routing time based upon subjective termination of the search once stagnation is detected.

Experiment 3: This experiment varied two mutation rates over a fixed range in incremental amounts while holding the other two mutation rates constant for the best results on problem M-n200-k17 in Experiment 2. The results of the experiment as shown in Figure 6 simply provide a visual display of the problem landscape as a projection of two dimensions. Portions of the surface near the front of the plot are not displayed if it were to block any part of the entire contour map, and consequently, cause data loss in the graph. Very small adjustments to mutation rates can impact the quality of solution discovered by GVR. This insight confirms the idea that the landscape searched is very jagged and that local minimal run rampant. Global minima are very difficult to find, even though the local minima are not very different in terms of percentage difference. For mission routing purposes, sensitivities to mutation parameters might be overcome by having several processors compute routes using different mutation parameters in parallel, or the mutators might be encoded as part of the chromosome in an effort to become self-tuning.

Experiment 4: Table 3 illustrates the results of running 11 different seeded runs with the best results from problem M-n200-k17. Segmentation packing produces better results in 6 of the 11 trials, but a Student's T-test with $\alpha = 0.05$ reveals that there is not a statistical significance between the two packing methods. Intuitively, it seems that the segmentation packing method should have produced superior results, because it does not disrupt the good building blocks.



(a) Displacement and Insertion



(b) Swap and Inversion

Figure 6: Effects of varying two mutation rates while holding the others constant.

A possible explanation that the two methods are not significantly different, however, might be that the recombination operators sufficiently exploit the solution space enough to compensate for any disrupted building blocks.

Experiment 5: Experiment 2 produced a solution of value 1348 for problem M-n200-k17 without reinitialization. Reinitializing with the same random number seed produced a solution of lower quality in all cases except for one, where the improvement was negligible. This indicates that 'good' partial solutions, when continually exploited, can continue to improve to some degree by converging to local minima. Reinitialization disrupts the good building blocks, even though it provides the opportunity for more of the search space to be examined. This experiment, like the previous ones, provides insight into the jaggedness of the search space and the abundance of local minima. A viable routing algorithm demands a quick high-quality solution, which is most likely not an optimum solution. Other approaches to improve solution quality should be tried, but in the interim, existing GVR solutions are still more than adequate for this particular application.

Experiment 6: The lambda exchange local search did not improve solution quality for any of the best results obtained from Experiment 2. The most likely reason is because the fitness landscape is especially treacherous for high quality solutions and the myriad local minima scattered throughout the space. It is uncertain whether another local search technique can improve near-optimal VRP solutions for the general case.

Table 3: Results from vehicle packings

Seed	Tight	Segmentation
1900	1385	1430
2003	1441	1445
2020	1424	**1348**
3131	1408	1387
4242	1387	1366
5353	1428	1360
5555	1420	1390
6038	1443	1387
7777	1444	1445
8765	1385	1394
9843	1389	1391
AVERAGE	1414	1394.818182
STD DEV	24.2363364	32.66440932

Table 4: Results from reinitialization.

	Seed 2020		Seed 3131		Seed 4242	
	4000	{8000,12000}	4000	{8000,12000}	4000	{8000,12000}
{5,10,15,20}	1359	1345	1412	1395	1361	1375

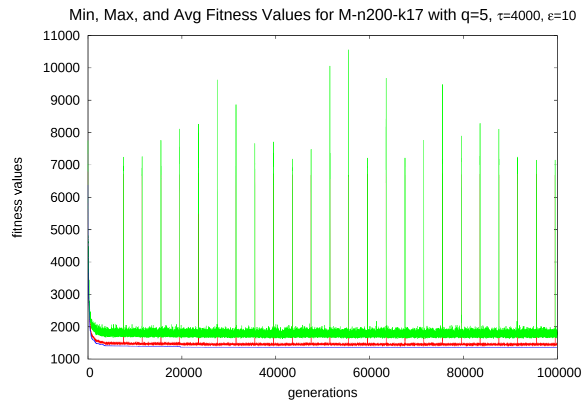


Figure 7: Reinitializations over time.

5. CONCLUSION & FUTURE WORK

High quality solutions for instances of the Capacitated Vehicle Routing Problem are obtained with Genetic Algorithms using the Genetic Vehicle Representation (GVR) encoding. A previously unmentioned aspect of the search using GVR–its convergence rate–is especially important to *a priori* and dynamic routing, and reveals that the GA exponentially increases solution quality. Another conclusion from statistical analysis is that a measure of control in the crossover and repair operators generally provides superior results than without it. GVR data encoding is also likely to perform competitively for VRP instances vulnerable to schedule changes while vehicles are en route because it exponentially increases solution quality. Real world applications of Unmanned Aerial Vehicle routing involve, at a minimum, the need for fast *a priori* routing schedules and mechanisms for scheduling changes en route. Analysis and experimentation validates that a GA using GVR encoding can provide these characteristics.

Future work involves extending GVR to solve Dynamic VRP instances as part of a online UAV routing system; multi-objective approaches should be considered. This research extends the Air Force Institute of Technology Swarm Simulator effort in support of the Air Force Research Laboratory Sensors Directorate (AFRL/SNZW), Wright-Patterson Air Force Base, Ohio.

The views expressed in this article are those of the authors and do not reflect the official policy of the United States Air Force, Department of Defense, or the U.S. Government.

6. REFERENCES

[1] Amit Agarwal, Meng-Hiot Lim, Maung Ye Win Kyaw, and Meng Joo Er. Inflight Rerouting for an Unmanned Aerial Vehicle. In *GECCO 2004*, 2004.

[2] P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, and É. D. Taillard. A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Research-C 5*, pages 109–122, 1997.

[3] T Bäck, D B Fogel, and T Michalewicz. *Evolutionary Computation 1*. Institute of Physics Publishing, 2000.

[4] T Bäck, D B Fogel, and Z Michalewicz. *Evolutionary Computation 2*. Institute of Physics Publishing, 2000.

[5] Branch and Cut. Vehicle Routing Data Sets.

[6] Joshua Corner. Swarming Reconnaissance Using Unmanned Aerial Vehicles In A Parallel Discrete Event Simulation. Master's thesis, Air Force Institute of Technology, 2004.

[7] Zbigniew J. Czech and Piotr Czarnas. Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows. *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, Canary Islands - Spain, (January 9-11, 2002):376–383, 2002.

[8] L.M. Gambardella, A. E. Rizzoli, F. Oliverio, N. Casagrande, A. V. Donati, R. Montemanni, and E. Lucibello. Ant Colony Optimization for Vehicle Routing in Advanced Logistics Systems. In *Proceedings of the International Workshop on Modelling and Applied Simulation (MAS 2003)*, pages 3–9, 2–3.

[9] D.E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[10] John H. Holland. *Adaptation in Natural and Artificial Systems*. Bradford Books, 1975.

[11] Jörg Homberger and Hermann Gehring. Two Evolutionary Metaheuristics for the Vehicle Routing Problem with Time Windows. *INFOR*, 37(3):297 – 318, August 1999.

[12] Mark Kleeman. Evaluation and Optimization of UAV Swarm Multi-Objectives. Master's thesis, Air Force Institute of Technology, 2004.

[13] John Levine and Frederick Ducatelle. Ant Colony Optimisation and Local Search for Bin Packing and Cutting Stock Problems. 2002.

[14] James Lotspeich. Distributed Control of a Swarm of Autonomous Unmanned Aerial Vehicles. Master's thesis, Air Force Institute of Technology, 2003.

[15] Penousal Machado, Jorge Tavares, Francisco B. Pereira, and Ernesto Costa. Vehicle Routing Problem: Doing it the Evolutionary Way. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, USA, 9-13 July 2002.

[16] Gunadi W. Nurcahyo, Rose Alinda Alias, Siti Maryam Shamsuddin, and Mohd. Noor Md. Sap. Vehicle Routing Problem For Public Transport A Case Study. 2000.

[17] Luiz S. Ochi, Dalessandro S. Vianna, Lucia M. A. Drummond, and Andre O. Victor. A Parallel Evolutionary Algorithm for the Vehicle Routing Problem with Heterogeneous Fleet. 1998.

[18] Peter Pae. Unmanned Aircraft Gaining The Pentagon's Confidence.

[19] Francisco B. Pereira, Jorge Tavares, Penousal Machado, and Ernesto Costa. GVR: a New Genetic Representation for the Vehicle Routing Problem. In *Proceedings of the 13th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2002)*, pages 95–102, Limerick, Ireland, 12-13 September 2002.

[20] T.K. Ralphs, L. Kopman, W.R. Pulleyblank, and L.E. Trotter Jr. On the Capacitated Vehicle Routing Problem. *Mathematical Programming*, Series B, 2003.

[21] JT Richardson, MR Palmer, G Liepins, and M Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In JD Schaffer, editor, *Proc. 3rd Int. Conf. on Genetic Algorithms*, pages 191–197. Morgan Kaufmann, 1989.

[22] Éric Taillard, Phillipe Badeau, Michel Gendreau, Francois Guertin, and Jean-Yves Potvin. A Tabu Search Heristic for the Vehicle Routing Problem with Soft Time Windows. Technical Report CRT-95-66, CRT, 1995.

[23] Jorge Tavares, Francisco B. Pereira, Penousal Machado, and Ernesto Costa. GVR Delivers It On Time. In *4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02)*, pages 745–749, 2002.

[24] Jorge Tavares, Francisco B. Pereira, Penousal Machado, and Ernesto Costa. Crossover and Diversity A Study about GVR. In *Proceedings of the Analysis and Design of Representations and Operators (ADoRo'2003) a bird-of-a-feather workshop at the 2003 Genetic and Evolutionary Computation Conference (GECCO-2003)*, Chicago, Illinois, USA, 12-16 July 2003.

[25] Jorge Tavares, Francisco B. Pereira, Penousal Machado, and Ernesto Costa. On the Influence of GVR in Vehicle Routing. In *Proceedings of the 2003 ACM Symposium On Applied Computing (SAC 2003)*, pages 753–758, Melbourne, Florida, USA, 9-13 March 2003.

[26] Sam R. Thangiah. An Adaptive Clustering Method using a Geometric Shape for Vehicle Routing Problems with Time Windows. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 536–543. Morgan Kaufmann, 1995.

[27] Chris Thornton. The Building Block Fallacy. *Complexity International*, 1997.

[28] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. SIAM, 2002.

[29] Matthew Walls. GAlib: A C++ Library of Genetic Algorithm Components.

[30] Darrell Whitley. A Genetic Algorithm Tutorial. 1994.

[31] Kenny Qili Zhu. A New Genetic Algorithm For VRPTW. In *International Conference on Artificial Intelligence, Las Vegas, USA*, 2000.