

Computational Intelligence Algorithms for Optimized Vehicle Routing Applications in Geographic Information Systems

Michael Rice
Thesis Proposal

Abstract

This project seeks to explore the application of two developing algorithmic paradigms from the field of computational intelligence towards optimized vehicle routing within geographic information systems (GIS). Ant Colony Optimization (ACO) is a type of multi-agent, swarm-based algorithm designed to mimic the emergent problem-solving behavior of real ants within a colony. Genetic Algorithms (GA) are another nature-inspired type of algorithm designed for evolving optimal or near-optimal solutions to a problem through the use of techniques based on natural selection, crossover, and mutation. The goal of this project is to implement and test a hybrid version of these two algorithms, aimed at evolving agents (ants) for optimized routing within the vehicle routing program.

Introduction

Within the last few decades of the 20th century, and continuing into the new millennium, there has been exponential growth in the research, development, and utilization of geographic information systems (GIS). However, while GIS have been designed to handle most types of common spatial analysis problems, many of the more complex spatial problems are still beyond their current capabilities to solve. These types of problems often involve extremely large search spaces with correspondingly large numbers of potential solutions. In such cases, standard analytical techniques typically fall short of finding optimal solutions to the problem within practical temporal and/or computational limits. One such problem within the realm of spatial analysis is that of the vehicle routing problem (VRP). This project will be focused on the development of algorithmic solutions for the VRP, designed specifically for solving such complex problems.

Vehicle routing has many practical applications within the fields of operations research, logistics, distribution, supply chain management, and transportation, to name a few. In general, vehicle routing involves finding efficient routes for vehicles along transportation networks, in order to minimize route length, service cost, travel time, number of vehicles, etc. As will be demonstrated in the next section, in which a more detailed and formal definition of the VRP is given, this is a combinatorial optimization problem for which no simple solutions exist. As an alternative, solution techniques from the field of computational intelligence shall be implemented and tested for solving instances of the VRP.

According to Engelbrecht (2002), computational intelligence may be defined as a sub-field of artificial intelligence (AI) that entails “the study of adaptive mechanisms to enable or facilitate intelligent behavior in complex and changing environments. These mechanisms include those AI paradigms that exhibit an ability to learn or adapt to new situations, to generalize, abstract, discover, and associate” (p.4). Computational intelligence is itself divided among many sub-fields of study, including (but not limited

to) artificial neural networks, evolutionary computation, swarm intelligence, and fuzzy logic. The two main fields of study that this project will focus on are swarm intelligence and evolutionary computation.

Swarm intelligence (SI) may be defined as “any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies” (Bonabeau, Dorigo, and Theraulaz, 1999, p.7). Many such algorithms have been developed to imitate the flocking of birds or the swarming of insects such as bees, ants, termites, etc. These algorithms are intended to make use of the emergent, self-organizing behavior of agents interacting locally within the swarm environment to optimize solutions to complex problems, such as function optimization, route finding, scheduling, structural optimization, and image and data analysis (Engelbrecht, 2002). The two most commonly researched algorithms from the field of SI are known as particle swarm optimization and ant colony optimization.

Similarly to SI, the field of evolutionary computation (EC) has developed as an attempt to aggregate several previously researched fields of related study into one, all-encompassing genre. The sub-fields of study include genetic algorithms, evolutionary programming, evolutionary strategies, and genetic programming. The common ground among all of these fields of EC is their use of computational techniques that are analogous to the evolutionary mechanisms that work within natural biological systems, such as natural selection (*i.e.*, survival of the fittest), crossover, mutation, etc. Within EC, these evolutionary mechanisms, or operators, are used as a means of quickly *evolving* optimal or near-optimal solutions to a problem within a computational framework designed to represent a relevant search space.

For this project, both ant colony optimization (ACO) and genetic algorithms (GA) shall be integrated into a newly-proposed, hybrid algorithm designed to optimize solutions to the VRP by evolving the perception and behavior of agents (ants) for route finding within the vehicle routing program. The hybrid algorithm will be implemented and tested on a variety of VRP instances. Other hybrid developments (*i.e.*, inclusion of local search operators and/or alternative heuristics) will also be tested for further improvements. The overall performance and solution qualities of all tested algorithms will be compared against each other, as well as against that of a widely used proprietary routing algorithm from the TransCAD GIS package.

The following sections of this paper are intended to give a more in-depth look at the VRP, as well as the ACO and GA techniques that will be used to solve it, including literature reviews of previous related research. A project setup will also be presented regarding the intended methodological approach, followed by the overall project objectives. Finally, a conclusion of the proposal will be made, with a discussion of the general expectations of the project, as well as a look at any potential problems or challenges that may be encountered.

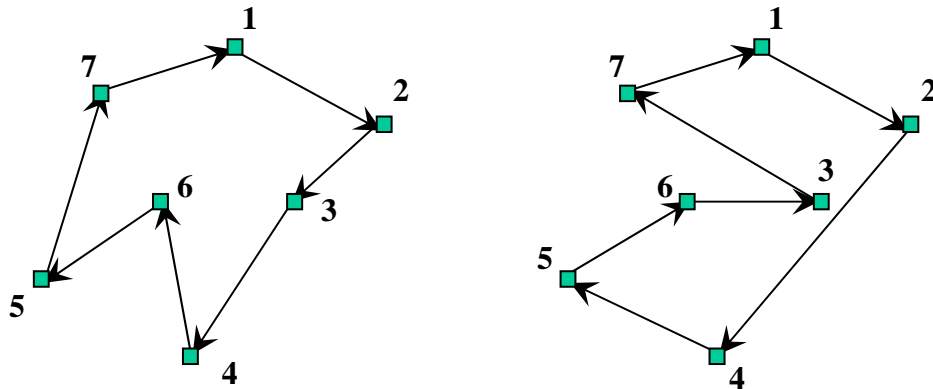
Vehicle Routing Problem (VRP)

In order to establish a proper foundation for discussion of the VRP, we must first take a look at a lower-level type of routing problem known as the Traveling Salesman Problem (TSP). The basic definition of the TSP is as follows. Given the cost of travel between n cities or stops, the objective is to minimize the total cost of a *single* route that visits all n cities/stops and returns to its starting point.

Intuitively, this may appear to be a seemingly trivial task, given the relative ease of computing the shortest possible path for single-stop routes using algorithms such as Dijkstra's algorithm or the Floyd-Warshall algorithm. However, any given TSP route is actually one of many possible *combinations* of *multiple* single-stop routes, thus making this type of multi-stop routing an NP-hard, combinatorial optimization problem.

The combinatorial aspect of TSP routing may be demonstrated in **Figure 1** below. In this graphic, two possible TSP routes for the same set of 7 cities/stops are created by using unique combinations of single-stop routes. Notice that it is the sequencing of stops that governs which single-stop routes are to be included in the overall TSP route.

Figure 1:



Given this situation, the number (R) of unique possible TSP routes for a given number (N) of cities/stops is equated as follows:

$$R = N!$$

However, if the starting city is fixed for all solutions, there are only $(N-1)!$ possible solutions. These formulations make TSP routing a problem that suffers from extreme combinatorial explosion. By this, it is meant that as the number of stops to be solved for increases, the problem becomes exponentially more complex.

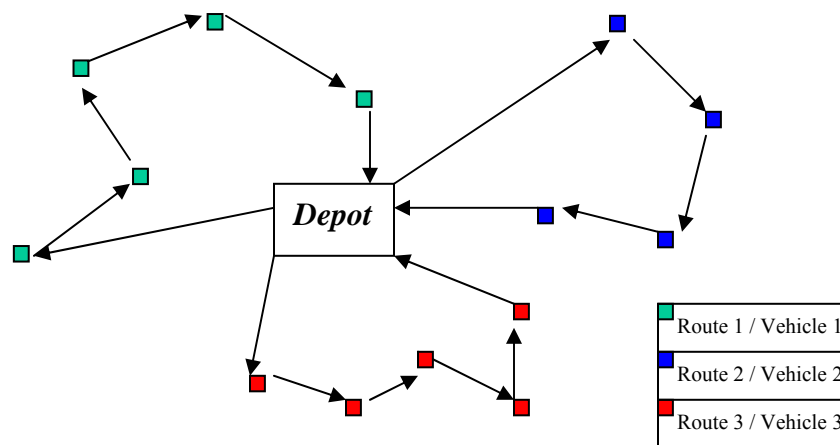
An example of the effects of combinatorial explosion on the search space of possible solutions for this type of routing problem may be shown in **Table 1** below. By noting the extremely huge increases in possible routes that accompany the small increases in stops to be made, the complexities of this problem should be made clear.

Table 1:

Number of Route Stops	Number of Possible Routes
5	120
10	3,628,800
15	1,307,674,368,000
20	2,432,902,008,176,640,000
30	2.65e+32
40	8.16e+47
50	3.04e+64

The VRP, on the other hand, is an even *more* complex, higher-level type of routing problem than that of the TSP, though the two problems are heavily related. The basic VRP involves determining an efficient set of *multiple* TSP routes, all starting and ending at a central depot, for a fleet of vehicles intended to service a given set of customers or stops. All customers or stops may be visited only once by only one vehicle, and, typically, no single route may exceed a given length or travel time, hence the need for multiple routes. As mentioned earlier, the objective of the VRP is to minimize route length, service cost, travel time, number of vehicles, or a combination of these, depending upon the particular application. A graphic representation of a routing solution to a VRP is presented in **Figure 2**, below, for comparison with standard TSP routing solutions, such as those shown in **Figure 1**^{*}.

Figure 2:



The VRP may actually be considered a broad *class* of routing problems, composed of many specific variants of the basic definition given above. Some of these VRP variants and their unique constraints are presented below:

1. *Capacitated Vehicle Routing Problem (CVRP)*- VRP with the additional constraint that all vehicles within the fleet have a uniform carrying capacity of a single commodity. The commodity demand along any route assigned to a vehicle must not exceed the capacity of the vehicle.
2. *Vehicle Routing Problem with Time Windows (VRPTW)*- VRP with the additional constraint that each customer or stop has an associated, fixed time interval during which pickups or deliveries must be made.
3. *Capacitated Vehicle Routing Problem with Time Windows (CVRPTW)*- VRP that includes both vehicle capacities and time windows (hybrid version of the CVRP and the VRPTW).
4. *Multiple Depot Vehicle Routing Problem (MDVRP)*- VRP with multiple depots and vehicle fleets. All stops must be assigned to a single depot/fleet in order to minimize service costs.

^{*} Note: all graphic representations of TSP and VRP solutions are depicted in Euclidean space (as opposed to actual network space) merely for the sake of simplicity and comprehension of the problem(s)

5. *Periodic Vehicle Routing Problem (PVRP)*- VRP that allows service to be extended over M days, instead of single-day service.
6. *Split Delivery Vehicle Routing Problem (SDVRP)*- VRP in which some customers may actually be serviced by more than one vehicle.
7. *Stochastic Vehicle Routing Problem (SVRP)*- VRP in which one or more problem components are random or present with some probability, p_i .
8. *Vehicle Routing Problem with Backhauls (VRPB)*- VRP in which both pickups and deliveries may be made at any given stop along a route. Typically, however, pickups are not made until after all deliveries along the route are made, hence the term backhaul.
9. *Vehicle Routing Problem with Satellite Facilities (VRPSF)*- VRP with the inclusion of remote facilities throughout the transportation network, which may be used to re-supply or unload vehicles along their route. This enables capacitated vehicles to handle routes with larger demands before returning to the central depot.
10. *Time Dependent Vehicle Routing Problem (TDVRP)*- VRP in which travel costs along the network are dependent upon the time of day during which travel is to be carried out. This variation is considered to be a more realistic approach, since traffic conditions are not typically static throughout the day.

The variant of the VRP that this project will be focused on is that of the CVRP. For this project, the CVRP, its representation, constraints, and objectives, may be defined more formally as follows (Bullnheimer, Hartl, and Strauss, 1999b):

CVRP representation-

- $G = (V, A, d)$ is a complete weighted directed graph
- $V = \{v_0, v_1, v_2, \dots, v_n\}$ is a set of vertices, such that v_0 represents the depot and all other vertices of V represent customer locations or stops to be made
- $A = \{(v_i, v_j) : i \neq j\}$ is a set of arcs, each with an associated, non-negative weight, d_{ij} , which represents the cost of traversing the arc (*i.e.*, length or travel time) between v_i and v_j
- For all v_i , a non-negative demand, q_i , is given ($q_0 = 0$)

CVRP constraints-

- Each customer or stop must be visited exactly once by exactly one vehicle
- All vehicle routes must begin and end at the depot
- All vehicles have capacity Q and route-cost limit L
- For every vehicle route, the total demand must not exceed Q and the total route cost (*i.e.*, length or travel time) must not exceed L

CVRP objectives-

- Obey all constraints
- Minimize the number of vehicles needed to service all customers/stops
- Minimize route costs for all vehicles

Because the CVRP is complicated by the inclusion of *multiple* instances of TSP routing, the problem of combinatorial explosion is even greater than for single TSPs. Brute-force algorithms designed to exhaustively search *all* possible solutions and determine the very best answer are impractical for CVRPs with large numbers of stops. With such huge numbers of possible solutions, exhaustive searching could take hours, days, months, or even *years* to find the best solution.

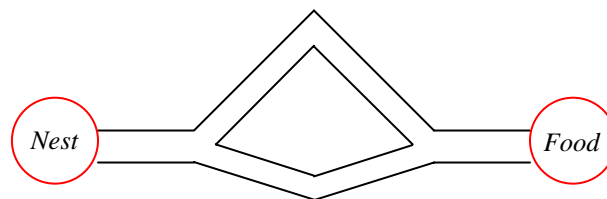
Therefore, *heuristics* and/or *metaheuristics* are typically employed to solve this type of problem. A heuristic is a logic-based algorithm, usually a general “rule-of-thumb”, designed to work quickly and provide very *good* solutions, though not necessarily always the *best* solution. A metaheuristic is a higher-level strategy that guides other heuristics in a search for improved solutions. This project will focus on the development of a hybrid metaheuristic algorithm for solving the CVRP, integrating techniques from both ant colony optimization and genetic algorithms. The following section introduces the ant colony optimization metaheuristic.

Ant Colony Optimization (ACO)

Ant colony optimization (ACO) is a multi-agent, metaheuristic algorithm designed to mimic the emergent behavior of real ants within a colony. Emergent behavior is exhibited when many low-level entities (agents) operate or interact within their environment to create a more complex, collective behavior. The ACO metaheuristic, formally developed by Dorigo, Di Caro, and Gambardella (1999), draws its inspiration from the experimental observations of emergent behavior in real ant colonies, such as the research and experimentation of Goss et al. (1989) on a laboratory-contained colony of Argentine ants.

Goss et al. designed experiments to test the foraging behaviors of ants within the colony by linking their nest to a remote food source by way of a branching bridge between the nest and food (as demonstrated below in **Figure 3**). The two branches of the bridge were of differing path length, and it was observed that, over a period of time, as the ants traversed the bridge in search of food, they began to converge upon the shorter of the two paths along the bridge. The convergence of the ants upon the shorter, more optimal path to the food source was an emergent behavior, as described above, of distinct path-finding ability resulting from a form of indirect communication known as *stigmergy* (Dorigo and Di Caro, 1999).

Figure 3:



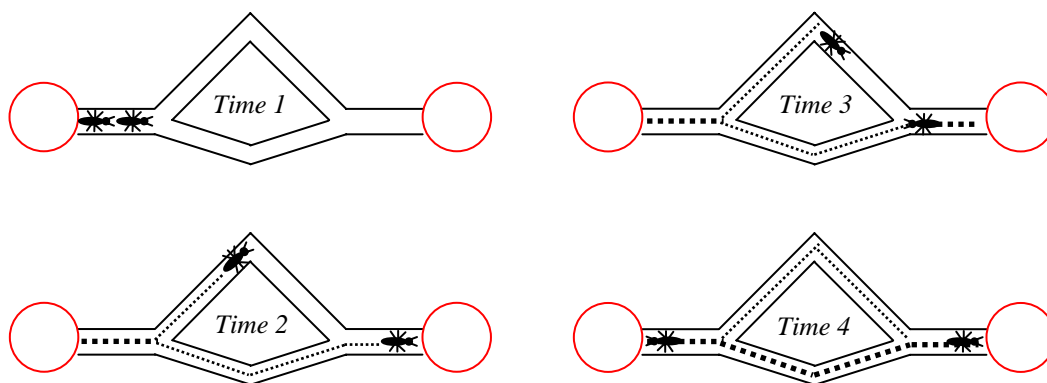
Stigmergy is defined as “a method of communication within decentralized systems, in which the individual parts of the system communicate with one another by modifying their local environment” (<http://en.wikipedia.org/wiki/Stigmergy>). In the case

of the ants foraging for food, the ants were able to modify their environment by depositing chemical signatures, known as *pheromones*, along the path of the bridge as they traveled. According to Dorigo and Di Caro (1999), by sensing the amount of pheromones deposited along branches of the bridge, each ant is able to make a probability-based decision, influenced by the amount of pheromone detected, over which branch to choose.

In the beginning, the ants would make random choices over which branch to traverse due to the initial lack of pheromones on the bridge. However, because of the differential path lengths of the two branches of the bridge, the ants choosing the shorter of the two branches will likely be the first to reach the food source. Upon leaving the food source to return to the nest, they will be able to detect pheromone trails along the shorter branch (*i.e.*, their old trail), and will be more likely to choose that path again for the return trip, consequently, further increasing the amount of pheromones along the shorter path (see **Figure 4** below). As more ants leave the nest in search of food, they will be more heavily influenced by the increasing amounts of pheromones along the shorter path.

Of course, this is a broad simplification of the process, as there are additional aspects that factor into this emergent path-finding ability. Over time, the pheromones deposited along a given path will begin to evaporate. Therefore, it is harder to maintain strong concentrations of pheromones along relatively longer paths, because it takes more time to traverse such longer paths and replenish pheromone concentrations than it does along the shorter paths. Due to the differential path lengths within a network of paths, more ants on average will be able to traverse the shorter of the paths, consequently increasing the disparity of pheromone levels between branches even further. All of these processes perpetuate themselves towards eventual convergence through what is known as *autocatalysis*, in which “the very fact of choosing a path will increase the probability that it will be chosen again by future ants” (Dorigo and Di Caro, 1999, p.1).

Figure 4: Dashed lines represent pheromone deposits. The weights of the lines represent relative intensity of pheromones.



The basic ACO metaheuristic is intended to replicate this emergent path-finding ability within networks represented as graphs, through the use of agents, known as

artificial ants, with similar capabilities to those of their natural counterparts. The agents, or ants, within the algorithm are able to cooperate to solve routing problems by locally sensing and depositing artificial pheromones along the edges of the graph.

Much like the many variants of the basic class of VRP (*i.e.*, CVRP, VRPTW, etc.), there also exist many specific variations on the basic ACO metaheuristic. Some of the more widely known variations of the ACO metaheuristic include Ant System (AS), Max-Min Ant System (MMAS), AS-Rank, Ant-Q, and Ant Colony System (ACS). The specifics of each of these variants are too detailed for the scope of this paper; however, the reader is referred to Dorigo, Di Caro, and Gambardella (1999) for a more in-depth discussion of each.

The specific version of ACO metaheuristic that this project will utilize for solving the CVRP is that of the Ant Colony System (ACS). ACS, a slight modification of the original Ant System (AS), was originally designed as an ACO algorithm for solving instances of the TSP, though it has also more recently been applied for solving instances of the VRP. For solving the more specific CVRP, the ACS may be implemented as follows.

Within the ACS algorithm, a colony of m artificial ants is used for determining an efficient set of routes to meet the overall objectives of the CVRP. ACS is designed to utilize the graph representation presented in the previous section for solving this type of routing problem. In addition to the cost weights, d_{ij} , associated with each network arc (v_i, v_j) in the set of arcs, A , the ACS supplements an additional “desirability measure”, τ_{ij} , which represents the amount of pheromone along the associated arc (v_i, v_j) (Dorigo and Gambardella, 1997, p.3). Upon initializing ACS, all arcs are given the same amount of initial pheromone τ_0 (usually a small constant). The pheromone values are then dynamically updated during the course of the ACS algorithm as the ants progress through the network searching for optimal routes.

According to Bullnheimer, Hartl, and Strauss (1999b), each ant within the colony builds feasible solutions to the problem by sequentially choosing stops $(v_j \in V)$ to visit, until *all* stops have been visited. Each solution begins from the central depot (v_0) and must return to the depot whenever the choice of another stop would result in a violation of vehicle capacity, Q , or route-cost limit, L , or when there are no more stops left to be visited. Upon each return to the depot, if there are still stops left to be visited, the ant must begin another route.

During route construction, each ant moves from one stop to another based on a certain set of selection criteria and probability-based rules. Each ant maintains a *tabu list* of stops already visited, so that it never visits any stop more than once in its overall solution. A stop is considered feasible for selection to move to next if it is not currently in the tabu list and moving to that stop would not violate Q or L .

Let $\Omega = \{v_j \in V : v_j \text{ is feasible to be visited}\} \cup \{v_0\}$ be the set of all feasible stops to be selected from at the current position v_i (Bullnheimer, Hartl, and Strauss, 1999b). In ACS, each ant selects the next stop to be visited in the routing sequence according to the *pseudo-random proportional rule*, formulated as follows (Dorigo and Gambardella, 1997):

$$s = \begin{cases} \max_{h \in \Omega} \left\{ [\tau_{ih}]^\alpha \cdot [\eta_{ih}]^\beta \right\} & \text{if } q \leq q_0 \quad (\text{exploitation}) \\ S & \text{otherwise} \quad (\text{exploration}) \end{cases}$$

where s is the stop selected to be visited next from the current stop v_i , τ_{ih} is the amount of local pheromones currently deposited along arc (v_i, v_h) , η_{ih} (known as *visibility*) is the reciprocal of the cost along arc (v_i, v_h) (i.e., $\eta_{ih} = 1/d_{ih}$), α and β are tunable parameters that determine the relative influence of pheromone trails and visibility, respectively, q is a real-valued random number $[0..1]$, q_0 is a tunable parameter ($0 \leq q_0 \leq 1$) that determines the amount of *exploitation* (of local information already present in the graph representation) vs. *exploration* (of other, possibly better route sequences) to be performed by the colony, and S is a random stop selected according to the probability distribution given in the following equation. This equation is known as the *random-proportional rule* (Dorigo and Gambardella, 1997):

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{h \in \Omega} ([\tau_{ih}]^\alpha \cdot [\eta_{ih}]^\beta)} & \text{if } v_j \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

where p_{ij} is the probability of selecting stop v_j while located at stop v_i . These rules, known as *state transition rules*, typically tend to favor arcs with lower costs and higher pheromone concentrations. However, the tunable parameters (α , β , and q_0) allow for the biases of these rules to be adjusted as needed.

While constructing routes using these selection/transition rules, each ant locally modifies the pheromones along the edges of the network as it travels. This is achieved by applying the following local updating rule:

$$\tau_{ij}^{new} = (1 - \rho) \cdot \tau_{ij}^{old} + \rho \cdot \tau_0$$

where $\rho \in (0, 1]$ is the pheromone decay coefficient and τ_0 is the initial pheromone deposit (a small constant). In this case, ρ represents the *persistence* of the pheromone and $(1 - \rho)$ represents the *evaporation* of the pheromone.

After each ant within the colony has constructed a feasible solution, pheromone trails are then updated *again* using the global-best routing solution, Ψ , found so far. This global updating rule is as follows:

$$\tau_{ij}^{new} = (1 - \rho) \cdot \tau_{ij}^{old} + \rho \cdot \Delta \tau_{ij}$$

$$\text{where } \Delta \tau_{ij} = \begin{cases} (C_{gb})^{-1} & \text{if } (v_i, v_j) \in \Psi \\ 0 & \text{otherwise} \end{cases}$$

and C_{gb} is the cumulative cost of all routes within Ψ .

As mentioned previously in this section, ACS was originally developed specifically for TSP applications. Dorigo and Gambardella (1996; 1997) and Gambardella and Dorigo (1996) have experimented with ACS for solving both symmetric and asymmetric instances of the TSP. Within their experiments, ACS was able to outperform (on average) other widely-used heuristics and computational intelligence applications implemented to solve the TSP, such as neural networks, simulated annealing, genetic algorithms, evolutionary programming algorithms, and the farthest insertion heuristic. The resulting “best set” of parameters that was empirically determined within these experiments is as follows: $m = 10$, $\beta = 2$, $\alpha = \rho = 0.1$, $q_0 = 0.9$,

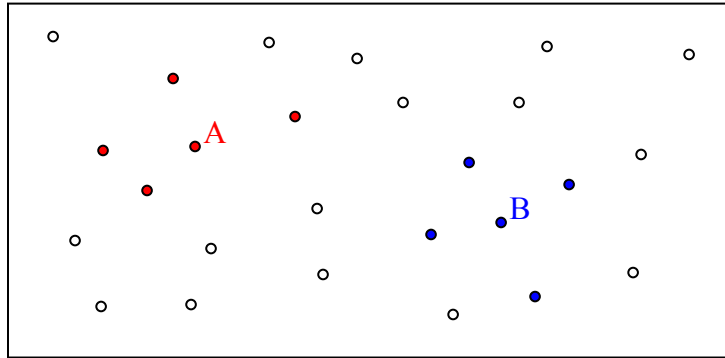
and $\tau_0 = (n \cdot C_{nn})^{-1}$, where C_{nn} is the cumulative cost of all routes in the nearest-neighbor (*i.e.*, greedy algorithm) routing solution and n is the number of stops to be solved for.

This research has been further advanced by Gambardella, Taillard, and Agazzi (1999) and Donati et al. (2003) to include applications of ACS to the VRP. Gambardella, Taillard, and Agazzi (1999) utilized multiple, cooperative ACS colonies, each designed to optimize a specific objective function, for solving instances of the VRPTW. Results across 56 VRPTW benchmark problems were compared with those of several other related studies and were found to be equally competitive with the best-known solutions. Donati et al. (2003) has more recently applied the ACS algorithm for solving the TDVRP, in which the researchers developed the concept of time-dependent pheromones by partitioning time into periods during which overall network speeds can be considered constant and associating differing sets of pheromones to each of these sub-space “time slices” (p.2). Results showed that this time-dependent approach proves quite valuable when applied to VRPs with dynamically changing network conditions.

Other highly relevant ACO research (besides that of ACS) applied to the field of VRP applications includes the work of Bullnheimer, Hartl, and Strauss (1999a; 1999b). The research of this group focuses on the heavily related Ant System (AS) algorithm for solving the VRP. One of the more significant contributions made by this group towards using ACO algorithms to solve VRPs is their use of *candidate lists* (further researched by Randall and Montgomery (2002)). A candidate list is a set of the cl nearest-neighbor nodes (in terms of cost) for every node within V (see **Figure 5**). During the route-construction phase of the AS algorithm, each ant is constrained to select the next node in the routing sequence from the candidate list, until all nodes within the candidate list are placed in the ant’s tabu list. The inclusion of candidate lists adds a degree of practicality to the selection process by reducing the possibility of including a highly undesirable (costly) move within a routing solution. The candidate lists were shown to improve both

algorithm run times and solution quality within the AS. The work of this research group was compared against several other computational intelligence applications, including tabu search, simulated annealing, and neural networks, in which the AS algorithm was able to outperform all but the tabu search method.

Figure 5: Candidate lists with $cl = 5$ for nodes A and B. The red nodes belong in node A's candidate list. The blue nodes belong in node B's candidate list.



The proposed research for this project intends to make use of the ACS algorithm augmented with the inclusion of a genetic algorithm and candidate lists for improving overall solution quality of the CVRP. The following section gives an introduction to genetic algorithms, as well as related applications of genetic algorithms for solving the VRP.

Genetic Algorithms (GA)

Genetic algorithms (GA) are the most fundamental and widely known form of evolutionary computation currently in research. The underlying concepts of the GA were originally developed by John Holland at the University of Michigan beginning in the 1960s while Holland was doing research in the field of complex adaptive systems. Subsequent research efforts by Holland, his university colleagues, and many other researchers have since advanced the GA into many practical forms of scientific application. These types of evolutionary algorithms have been designed as models capable of solving decision problems, classification problems, and complex numerical optimization problems (Krzanowski and Raper, 2001).

More specifically, GAs are search heuristics designed to work through the use of computational techniques based on natural selection, crossover, and mutation. In order to understand how a GA may be implemented as a computational procedure for finding a solution to a problem, the following steps are presented, complete with proper bio-metaphoric terminology, as the basic makeup of a typical genetic algorithm (Krzanowski and Raper, 2001):

- **Step 1:** Initialize a *population*, composed of individual *chromosomes* → this creates multiple potential solutions to the problem (often randomly instantiated)
- **Step 2:** Evaluate the *fitness* of each *chromosome* → this rates how well a potential solution achieves the objective function of the problem

- **Step 3:** Select *chromosomes* for *crossover* based on their relative *fitness* → this ensures only the fittest get selected to breed the next *generation* of potential solutions
- **Step 4:** Perform *crossover* of selected *chromosomes* → this produces a new *population* of potential solutions that maintain certain attributes of their selected parents
- **Step 5:** Perform *mutation* on *chromosomes* → this introduces new random variation(s) into the *population* in order to more widely explore the search space
- **Step 6:** Evaluate the *fitness* of each *chromosome* in the new *population* → after successive *generations* of *populations*, the overall *fitness* is expected to increase (theoretically, until the objective function is met)
- **Step 7:** If the objective function has been met within the new *population*, **STOP**; otherwise, repeat from **Step 3**

The objective function to be met by the genetic algorithm is dependent upon the particular problem domain. Genetic algorithm optimization generally includes minimizing or maximizing this objective function.

One of the most important aspects of genetic algorithm design is that of chromosome representation (*i.e.*, the representation of potential solutions in the algorithm). The GA chromosome is mapped to a set of relevant parameters within the problem definition. Most classical genetic algorithms make use of a binary (*i.e.*, bit-string) representation. However, more recent versions have also made use of integer- or real-valued representations.

Using a problem-dependent fitness function, each chromosome (solution) may be assigned a relative *fitness* value based on how well it achieves the objective function of the given problem. Given these fitness values, solutions are then evolved from generation to generation using the three main genetic operators previously mentioned: *selection*, *crossover*, and *mutation*.

Selection is the method by which individual chromosomes within a population are chosen for crossover. This selection is to be based on the fitness values of individual chromosomes. The better the fitness of the individual chromosome, the better the chances are for its selection. There are several different types of selection operators that may be incorporated into a genetic algorithm, the two most common of which will be discussed here.

Roulette Wheel Selection- in this method, each chromosome is selected with a probability proportional to its percentage of the total fitness of the population. This operator is analogous to spinning a roulette wheel and choosing the division of the wheel that is landed on when the wheel stops spinning. Each individual occupies a division of the wheel proportionally sized to its relative fitness within the population as a whole. Thus, individuals with very high fitness have very high probability of being selected many times. However, this oftentimes decreases the overall diversity of the population.

Tournament Selection- in this method, selection is determined by *k*-sized tournaments, in which *k* number of chromosomes are randomly chosen from the population and evaluated against all other chromosomes chosen for the given tournament. The best-fit solution from each tournament is selected for crossover. This selection method does not necessarily always favor the most-fit individual *of an entire population*.

Once individual chromosomes have been selected, they may then be passed to the crossover operator. Crossover is the method by which individual chromosomes within a population are mated together to form new individuals to populate the next generation. The newly created individual chromosomes will typically maintain certain features from *both* of their parent chromosomes. Crossover is usually applied based on a pre-determined probability p_c . The two most commonly used crossover operators are presented below.

One-Point Crossover- this crossover method may be applied as follows, given two bit-string chromosomes (parent chromosomes) selected from a given population:

Parent A: 010010
Parent B: 110111

A single random crossover point is chosen between adjacent genes of both parent chromosomes (as illustrated above), and the genetic material on either side of the cross point is swapped between parents, resulting in two new children chromosomes (as shown below):

Child C: 010111
Child D: 110010

Two-Point Crossover- this method is very similar to the one-point crossover method mentioned above, except that two cross points are determined instead of one,

Parent A: 010010
Parent B: 110111

and the genetic material *in between* the two cross points is swapped between parents. In essence, this treats the chromosome as if it were a ring of genes.

Child E: 010110
Child F: 110011

After crossover has been performed, random mutations are often introduced based on another pre-determined probability p_m . Upon mutation, a randomly chosen gene within a chromosome will be replaced with another randomly chosen gene.

Before mutation: 010110
After mutation: 011110

Mutation is introduced as a random movement through the search space in the hopes of providing more diversity within the population. In most instances of genetic algorithm use, mutation is typically seen as more of a secondary search operator when compared to that of crossover, which is usually intended to perform the bulk of the directed search.

Genetic algorithms have been numerous researched and developed as both standalone and hybrid solutions to multiple TSP and VRP application areas (far too many times, in fact, to create a comprehensive reference list of such broad research). However, this research focuses solely on the hybrid integration of genetic algorithms with ACO metaheuristics for solving related routing problems. Some of the more relevant (and recent) related research of ACO/GA hybrids includes the work of Botee and Bonabeau (1998) and Pilat and White (2002).

Botee and Bonabeau (1998) have made use of a genetic algorithm for evolving multiple parameters of the ACS algorithm when applied to TSP routing. The parameters evolved included m , α , β , ρ , q_0 , and τ_0 , as well as 5 additional implementation-specific parameters developed by Botee and Bonabeau. The GA was designed with a bit-string chromosome representation, in which each gene (parameter) was encoded as a 16-bit section of the overall chromosome. Each 176-bit chromosome represented an entirely separate colony of ants. The fitness function was based on the route solution qualities obtained by each colony using the ACS algorithm with their associated chromosomal parameters. The GA was ran with a population size of 40 (chromosomes/colonies) for up to 100 generations.

When compared to earlier benchmark experiments of hand tuning these parameters, the hybrid approach resulted in increased convergence speed, as well as highly comparable solution qualities. Though this hybrid approach proved very competitive with standard ACS implementations, it was also very computationally intensive due to the fact that it must run the ACS on 40 different colonies in parallel *per generation*. Preliminary experiments on the use of a GA to evolve a division of labor, or caste system, among the ants within the ACS algorithm were also discussed and showed promising potential for future studies.

Pilat and White (2002) have developed a similar use of GAs within an ACS implementation for solving TSP routing. However, they have taken two differing approaches towards the integration of a GA within their system. One of the experimental approaches was very much like that of Botee and Bonabeau (1998), as described above, except that only three different parameters (β , ρ , and q_0) were encoded as bit-string chromosomes, and only 4 bits per parameter were used to help reduce the search space. Again, each chromosome was representative of an entire colony, thus making for another quite computationally intensive approach. Results from this experiment yielded different parameter values, depending upon the particular routing problem being solved for. This has led these researchers to believe that, unlike the universal “best set” of parameters promoted by Dorigo and Gambardella (1997) (as mentioned in the previous section), the tunable ACS parameters may actually be quite unique to the given problem.

The second experiment within this research took on a wholly different approach by genetically encoding *individual ants* with their own set of parametrical chromosomes, to be evolved within a single colony. Therefore, instead of using a global parameter set for the entire colony, each ant was now able to make uniquely biased decisions within the ACS algorithm. Each ant was encoded with a bit-string chromosome of 21 bits (7 bits per parameter), designed to represent the three parameters evolved in the above experiment (β , ρ , and q_0). Using tournament selection (tournament size = 4), 4 ants were selected from the colony and allowed to build routing solutions while locally updating the

pheromone trails. The ant producing the best routing solution was allowed to again globally update the pheromone trails. The two ants from the 4-ant tournament with the highest-ranking fitness were then crossed over and mutated to produce two children to replace the two ants from the tournament with the lowest ranking fitness. The four ants were then placed back in the population for another round of selection, crossover, and mutation. Unlike standard GA methods, which replace the entire population with a newly derived population each generation, this method maintains a constant population in which only a few chromosomes are replaced at a time. This method is known as a *steady-state GA*. The results of this experiment were compared to that of a standard ACS algorithm over 4 different benchmark problems. Compared to the standard ACS, the hybrid ACS resulted in an increased convergence speed and slightly higher quality routing solutions (on average). It is noted by these researchers that the results of this latter experiment could possibly be improved by a more complex genetic algorithm within the ACS.

For this project, it is proposed to develop such a hybrid improvement through the integration of a genetic algorithm within the ACS framework. The proposed algorithm will make use of a colony of genetically encoded ants for optimizing routing solutions to the CVRP (as opposed to merely the TSP routing in the above research). Each ant will be encoded with its own genetic chromosome, which is mapped to a set of tunable parameters within the ACS algorithm, giving it a unique set of decision-making capabilities for route finding. The algorithm will then evolve the colony of ants towards an optimal or near-optimal routing solution to the given problem.

Within this proposed hybrid algorithm, it is believed that the inclusion of a real-valued chromosome representation will make for a much more natural and robust encoding of the parameters than that of the previously used binary representation. This is based on the fact that the parameters themselves are real-valued, and this information is quite easily disrupted when randomly manipulating combinations of representative bits.

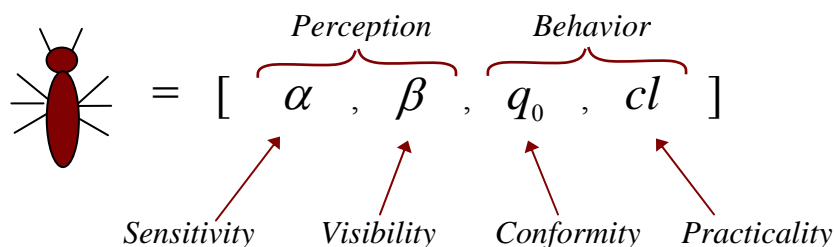
It is also believed that a much better choice of parameters to be evolved can be made in light of the previous research. A good choice of parameters to be evolved within this proposed algorithm should be as specific to the individual ants as possible, without adversely affecting the performance of the other ants within the colony. The inclusion of the evolvable parameter m , as was done in Botee and Bonabeau (1998), was completely colony-specific and is therefore only possible when evolving entire colonies of ants, while the inclusion of the parameters τ_0 and ρ , as was done by Botee and Bonabeau (1998) and Pilat and White (2002), respectively, may have unwanted negative influences upon the decisions of other ants within the colony regarding their effects upon the pheromone trails (especially when encoded at the level of the individual ant). Therefore, the proposed set of ACS parameters to be genetically encoded within the ant chromosomes is as follows: α , β , and q_0 . All other tunable ACS parameters (m , ρ , and τ_0) will be empirically determined and set as constants at run time.

Additionally, this project seeks to incorporate candidate lists within the hybrid algorithm, in the hopes of improving both running times and solution qualities even further. The difficulty in empirically determining an optimal candidate list size, cl , for a given routing problem is much the same as determining the optimal set of tunable parameters for the ACS. Similar to Pilat and White's belief that the optimal set of tunable parameters for the ACS is truly dependent upon the particular problem being solved for, it is also believed that the optimal size of the candidate lists may be unique to the

particular problem as well. According to Randall and Montgomery (2002), a dynamic candidate list strategy (if properly implemented) will likely lead to further improvements over the original, static candidate list strategy, in which cl is defined *a priori*. It is hoped that the inclusion of cl within the ant chromosome will help further the ACS system capability.

The full genetic encoding of an individual ant within the system will then be: α , β , q_0 , and cl . These parameters determine the decisive influence of pheromones, visibility, exploitation vs. exploration, and scrutiny of nearest-neighbor nodes within the graph, respectively. Therefore, the phenotypic traits that this proposed genotype will express will be equated to the sensitivity of the ant to pheromones, the visibility of the ant, the conformity of the ant, and the practicality of the ant, again respectively. The first two of these phenotypic traits make up the ant's perception, while the last two traits make up the ant's behavioral personality. The proposed chromosome is demonstrated below in **Figure 6**.

Figure 6: *The proposed genetic encoding of an ant chromosome within the ACS algorithm, including the expressed phenotypic traits.*



As demonstrated in the above graphic, in the case of the ACS/GA hybrid, the “solution” to be evolved by the genetic algorithm is the set of transition rule parameters, which, ultimately, influences the ants’ overall decision making capabilities when route finding. The inclusion of the genetic algorithm within the ACS algorithm will therefore operate as follows.

Upon initialization of the algorithm, each ant’s chromosome will be instantiated with a randomly created set of real-valued numbers (each within the allowed range of parameter values), as demonstrated below for a single ant within the colony:

$$\text{Ant} = [0.6, 2.8, 0.75, 5]$$

Using the steady-state GA scheme used by Pilat and White (2002) (discussed earlier in this section), a tournament of 4 ants will be randomly selected from the colony and allowed to build routing solutions while locally updating the pheromone trails. During this phase of the algorithm, the ants will build their routing solutions by simply plugging their individual chromosome values into the associated parameters within the state transition rules, and choosing successive stops along their routes as indicated by the results of these rules. The ant producing the best routing solution will be allowed to again globally update the pheromone trails. The two ants from the 4-ant tournament with the

highest-ranking fitness (which is based on the cost of their routing solution) will then be crossed over (based on a pre-defined crossover probability) to produce two new children (as demonstrated in **Figure 7**). Each of these two new children will then be subject to mutation (based on a pre-defined mutation probability), which would slightly alter their chromosome values to further explore the search space (as demonstrated in **Figure 8**).

Figure 7: Crossover of two ant chromosomes in the ACS/GA algorithm. A crossover point is randomly chosen. The two parents' chromosomes (top) are then crossed over to create two new child chromosomes (bottom).

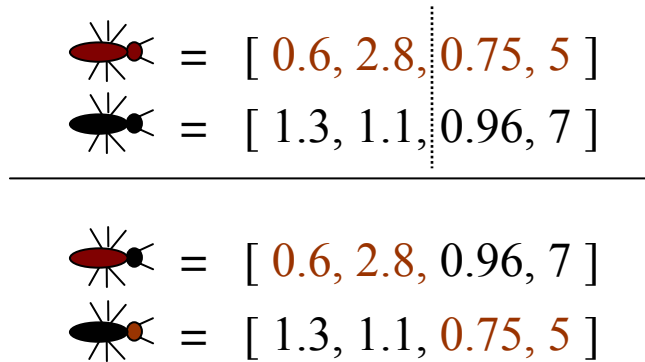
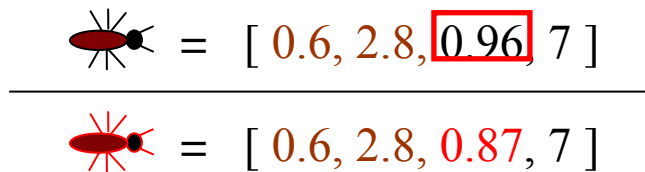


Figure 8: Mutation of an ant chromosome in the ACS/GA algorithm. A random gene is chosen within the ant's chromosome (top), and its value is replaced by a new randomly chosen value (bottom).



These two new children would then replace the two ants from the tournament with the lowest ranking fitness. The four ants (the two parents and their two new children) will then be placed back in the population again for another similar iteration, until some specified termination criteria is met, such as a maximum number of iterations, a threshold solution quality, a threshold of population diversity, etc.

Using this scheme, the hybrid ACS/GA algorithm will evolve both the perception and behavior of ants within the vehicle routing program, in order to optimize their decision base for route finding and, ideally, the overall solution quality of the algorithm. The following section will present the project setup, designed to test the overall capability of this proposed algorithm for the CVRP.

Project Setup

According to the latest vehicle routing software survey in the magazine *OR/MS Today* (2004) (*OR/MS* stands for *Operations Research/Management Science*), several of

the algorithms employed by many mainstream transportation GIS software companies for vehicle routing include genetic algorithms, clustering algorithms, tabu search, guided local search, hybrid heuristic algorithms, and various other proprietary implementations based on past related research of the VRP.

However, during the research of this proposal, it has been continually noted that rarely, if ever, have any of the researched algorithmic implementations for the VRP been compared against these more mainstream VRP solution applications. This research will not only focus on the comparisons between algorithms developed during the course of this project, but it will also seek to make relevant comparisons with that of the proprietary vehicle routing algorithm from the widely used and well-known TransCAD GIS package.

At least 10 problem instances of the CVRP will be (arbitrarily) created within TransCAD to be used for the evaluation of the newly proposed hybrid algorithm. These problem instances will be successively created in levels of increasing complexity (*i.e.*, ranging from 10 stops to 100+ stops). These problems will be solved using a standard ACS algorithm implementation, the newly proposed hybrid ACS/GA algorithm implementation, and TransCAD's own proprietary vehicle routing algorithm. Comparison against solutions from such mainstream GIS software will allow for a better gauging of the overall project success.

Project Objectives

The overall objectives of this project are as follows:

- Develop and test a newly-proposed, hybrid vehicle routing algorithm, integrating techniques from the fields of swarm intelligence and evolutionary computation (as outlined above)
- Test for other hybrid improvements within the implemented algorithm (*i.e.*, inclusion of local search operators and/or alternative heuristics)
- Compare results of the developed algorithm(s) against a currently-existing, mainstream proprietary vehicle routing algorithm (such as that used in TransCAD), as well as against the standard ACS algorithm implementation
- From the results, draw conclusions on the problem dependency of the ACS parameters and candidate list size
- Discuss any implementation-specific details important to future GIS development regarding solutions to the VRP (*i.e.*, run time analysis and comparisons, deterministic approaches versus probabilistic approaches, user interface, user expectations, ease of development, etc.)

Conclusion

This proposal has presented an in-depth look at the vehicle routing problem (VRP), as well as some relevant research within the field of computational intelligence aimed at solving the VRP. A newly proposed hybrid algorithm solution to the VRP was presented, integrating techniques from both the field of swarm intelligence and the field of evolutionary computation.

This algorithm shall be tested on multiple capacitated vehicle routing problems designed within the TransCAD GIS software package, and compared against several other solution applications. Based upon the representational nature of the proposed

algorithm, it is believed that this algorithm may be able to improve upon the standard Ant Colony System implementation, as well as the implementations of previously attempted hybrid approaches. Though there is always the possibility of an unsuccessful endeavor, it is imperative that this newly proposed methodology be fully tested, in order to gain a better assessment of its true capabilities and limitations.

References

- Bonabeau, E., M. Dorigo, and G. Theraulaz (1999) From Natural to Artificial Swarm Intelligence. New York: Oxford University Press.
- Botee, H.M. and E. Bonabeau (1998) "Evolving Ant Colony Optimization". In Advanced Complex Systems, 1:149-159.
- Braysy, O. (2001) "Genetic Algorithms for the Vehicle Routing Problem with Time Windows". Arpakannus 1, pp.33-38 (special issue on Bioinformatics and Genetic Algorithms).
- Bullnheimer, B., R. F. Hartl, and C. Strauss (1999a) "Applying the Ant System to the Vehicle Routing Problem". In: S.Voss, S. Martello, I.H. Osman, and C. Roucairol (Eds.) Metaheuristics: Advances and Trends in Local Search for Optimization, Kluwer Academic Publishers, Boston, 285-296.
- Bullnheimer, B., R.F. Hartl, and C. Strauss (1999b) "An improved ant system algorithm for the vehicle routing problem". Annals of Operations Research, 89:319-328.
- Donati, A.V., et al. (2003) "Time Dependent Vehicle Routing Problem with an Ant Colony System". In: internal IDSIA report, IDSIA-02-03, 20-Jan-03.
- Dorigo, M. and L.M. Gambardella (1996) "Ant colonies for the traveling salesman problem". Universit'e Libre de Bruxelles, Belgium, IRIDIA, Technical Report TR/IRIDIA/1996-3.
- Dorigo, M. and L.M. Gambardella (1997) "Ant Colony System: A cooperative learning approach to the Traveling Salesman Problem". IEEE Transactions on Evolutionary Computation 1(1): 53-66.
- Dorigo, M. and G. Di Caro (1999) "The Ant Colony Optimization Meta-Heuristic". In: D. Corne, M. Dorigo, and F. Glover (Eds.) New Ideas in Optimization. McGraw-Hill, 11-32.
- Dorigo, M., G. Di Caro, and L.M. Gambardella (1999) "Ant Algorithms for Discrete Optimization". Artificial Life 5(2): 137-172.
- Engelbrecht, A.P. (2002) Computational Intelligence: An Introduction. John Wiley & Sons, Ltd.
- Gambardella, L.M. and M. Dorigo (1995) "Ant-Q: A Reinforcement Learning approach to the traveling salesman problem". Proceedings of ML-95, Twelfth International Conference on Machine Learning, Tahoe City, CA, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann, 252-260.

Gambardella, L.M. and M. Dorigo (1996) "Solving symmetric and asymmetric TSPs by ant colonies". Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96, pp. 622-627. IEEE Press.

Gambardella, L.M, E. Taillard, and G. Agazzi (1999) "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows". In: D. Corne, M. Dorigo, and F. Glover (Eds.) New Ideas in Optimization. McGraw-Hill, London, UK, pp. 63-76.

Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization, & Machine Learning. New York: Addison-Wesley.

Goss, S., et al. (1989) "Self-organized shortcuts in the Argentine ant". Naturwissenschaften, 76: 579-581.

Gunay, A. (2003) "Using GAACO for the Solution of TSP and QAP". The IEEE Computer Society's Student Magazine, Fall 2003, 10(3): 4-8.

Guntsch, M. and M. Middendorf (2001) "Pheromone modification strategies for ant algorithms applied to dynamic TSP". In: E.J.W. Boers et al., editor, Applications of Evolutionary Computing: Proceedings of EcoWorkshops 2001.

Guntsch, M. and M. Middendorf (2002) "A population based approach for ACO". In: S. C. et al., editor, Applications of Evolutionary Computing - EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN, number 2279 in Lecture Notes in Computer Science, pp. 72-81. Springer Verlag.

Krzanowski, R. and J. Raper (2001) Spatial Evolutionary Modeling. New York: Oxford UP.

Pilat, M.L. and T. White (2002) "Using Genetic Algorithms to Optimize ACS-TSP". Proceedings of Ant Algorithms: Third International Workshop, ANTS 2002, Brussels, Belgium, September 2002, M. Dorigo et al (Eds.), LNCS 2463, Springer-Verlag, 282-287.

Randall, M. and J. Montgomery (2002) "Candidate set strategies for ant colony optimization". Ant Algorithms. pp. 243-249.

Vehicle Routing Software Survey (2004). *OR/MS Today*, June.