LOW-THRUST TRAJECTORY OPTIMIZATION USING STOCHASTIC OPTIMIZATION METHODS

BY

JOHN WILLIAM HARTMANN

B.S., University of Illinois at Urbana-Champaign, 1996

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Aeronautical and Astronautical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1999

Urbana, Illinois

For Dad

Acknowledgements

I would like to thank my advisor, Dr. Victoria Coverstone-Carroll for all of her encouragement and support during the course of my research. Her insights and advice were invaluable to the production of this thesis.

I would also like to thank Steve Williams for the guidance he provided in directing the research efforts endeavored upon.

I would like to thank the fellow members of my research group who put up with many a long and boring presentation, and especially Bill Mason whose cooperative efforts were much appreciated.

Most of all I would like to thank my family and friends for supporting me through the course of my education. Their encouragement and pride in my accomplishments has driven me to succeed at things I might not otherwise have attempted.

Table of Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

This thesis outlines a method for the optimization of low-thrust, interplanetary, spacecraft trajectories. In particular, it describes trajectory optimization through the use of stochastic optimization algorithms. The two most widely recognized stochastic methods, simulated annealing and genetic algorithms, are utilized. In this introductory chapter, background is provided to motivate the topic, and the objectives of this study are presented. An overview of the structure of the thesis is also provided.

1.1 Background and Objectives

The following sections provide background on the subject of trajectory optimization using stochastic optimization algorithms. This is accomplished through a review of current literature dealing with similar or related topics. The reader will readily note that current research on trajectory optimization using stochastic methods almost exclusively involves the application of genetic algorithms. This is interesting in the foreshadowing it provides with regard to the eventual course of this thesis. Also worthy of note are the several studies included in the survey that implement a hybridized algorithm. Following this review, current research efforts motivating the use of stochastic methods for low-thrust trajectory optimization are provided.

1.1.1 Stochastic Methods in Astrodynamics

During the last several years, investigators have used genetic algorithms (GAs) to optimize various aspects of complex space missions. These missions are diverse in nature and range from trajectory planning for launch vehicles to the design of interplanetary missions. For example, a guidance method for the ascent trajectory of a single stage to low-earth-orbit (LEO) launch vehicle has been developed with the aid of a genetic algorithm.[1] In this method, an optimal nominal trajectory is first determined prior to launch. A linear-quadratic-regulator-based guidance scheme is employed to keep the vehicle on course. Whenever the actual vehicle deviates from the nominal path by more than a prescribed margin, a new optimal trajectory is determined for the remainder of the flight. This "on-line" re-optimization is provided by a gradient-based optimization algorithm.

It is in the optimization of the nominal trajectory that a genetic algorithm comes into play. A GA is used to perform this "off-line" trajectory optimization. Trajectories are generated using a multiple-node cubic spline technique. The parameters optimized by the GA are the initial slope of the trajectory and the altitude values defined by the cubic splines at each trajectory node. The optimization is performed with the objective of maximizing the payload delivered to LEO. Each trajectory is evaluated by integrating the state equations forward until the final time and finding the payload to LEO. In addition, a single constraint associated with the maximum dynamic pressure the vehicle is capable of withstanding is also applied. Final results of the method used were compared with the high-fidelity optimization algorithm, Optimal Trajectories Implicit Simulation (OTIS). The performance difference was found to be within 2% of the OTIS-predicted result. The resulting GA guidance method provided good performance, with computational costs lower than traditional methods.

The feasibility of using a genetic algorithm for the optimization of an ascent trajectory from the surface of the moon is investigated Reference 2. The genetic algorithm generates a thrust profile for each trajectory with the overall objective of minimizing the flight time for a given terminal orbit. The GA results were compared with an analytical solution and were also measured against results obtained by a calculus-based optimization method. Both comparisons were favorable. During the study, several local optima were found to exist. A hybrid methodology was developed where the GA is used to determine a nominal trajectory, and the calculus-based method then optimizes the GA-derived solution. The hybrid technique outperformed the GA alone. The study concluded that genetic algorithms are a feasible method for trajectory optimization.

Another application of GAs for the purpose of mission planning entails sending spacecraft to rendezvous with comets, landing on their surface and taking samples.[3] A GA was developed to determine a complex set of maneuvers to fly over five candidate landing sites. The objective was to minimize the propellant required by the chemical propulsion system during the fly-over of the five candidate landing sites. The maneuvering strategy is complicated by the existence of multiple nonlinear practical constraints. A gradient-based algorithm previously developed for the purpose of determining fly-over maneuvers was found to be incapable of dealing with the posed problem due to the number of fly-over sites and complicated constraints. The GA, however, was able to provide several unique and highly efficient solutions.

Traditionally, interplanetary space missions have employed chemical high-thrust propulsion systems to perform the necessary energy exchanges that are required to complete the mission objectives. During preliminary design, the energy exchange is assumed to occur instantaneously so that the optimization process must determine the moment in time that the engines are activated, and the magnitude and direction of the velocity of the spacecraft after the engines have fired. Using such a model, a one-way Earth to Mars trajectory was studied in Ref. 4 and the associated parameter space was found to be multi-modal and discontinuous. The objective was to minimize the initial mass in LEO that would deliver a 6000 kilogram spacecraft to Mars. The Earth departure date and Mars arrival date were allowed to vary. A GA was successful in determining a global and several locally optimizing solutions.

As demonstrated with the October 24, 1998 launch of the first low-thrust propulsion interplanetary space mission, New Millennium Deep Space 1 (DS1), changing philosophies in mission design have produced renewed interest in low-thrust propulsion systems. Such low-thrust systems, however, complicate trajectory optimization due to the fact that the magnitude and direction of the propulsion system thrust is a continuous function of time. Refs. 5 and 6 use macro and micro GAs to determine the magnitude and direction at discrete locations along the trajectories for such low-thrust trajectory optimization problems. Mission objectives considered include minimizing the amount of propellant required to perform the mission or minimizing the entire mission length. Mission constraints are enforced as penalties to the objective function. Results were shown to closely match published results that were determined using gradient-based optimization methods.

Genetic algorithms have also been a useful tool for determining near-optimal low-thrust spacecraft trajectories in instances when applying second-order necessary and sufficient conditions within a calculus of variations framework fail.[7,8] A simple GA coupled with an ordinary differential equation integrator for trajectory propagation is used for trajectory generation. The GA determines the control parameters for a trajectory, and the system equations of motion are propagated forward using the integration routine to determine the final spacecraft state. Fitness is assigned according to the degree that the desired final spacecraft state was met. Terminal constraints are applied via appended penalty functions.

The utilization of simulated annealing (SA) in spacecraft mission design has not been developed to this point. While optimization using simulated annealing algorithms has been attempted in the areas of spacecraft design, actuator placement, and geostationary orbit control,[9,10,11] to the author's knowledge, SA algorithms have not been employed for optimization of spacecraft trajectories, low-thrust or otherwise. Thus no precedent has been established for the use of simulated annealing in spacecraft trajectory optimization.

1.1.2 Low-thrust Trajectory Optimization

In recent years, pressure to reduce the costs of interplanetary missions has led to a heightened emphasis on designing missions with shorter flight times, smaller launch vehicles, and simpler flight systems. These requirements have renewed interest in low-thrust propulsion systems due to their high propellant efficiencies; however, these propulsion systems require a different dynamic model than their high-thrust counterparts due to the fact that the energy exchange imparted to the spacecraft occurs over a finite period of time. Typically, low-thrust missions require the propulsion system to operate for a significant percentage of the entire mission length in order to generate the energy levels required by mission objectives. Trajectory optimization involving low-thrust systems is thus complicated by the fact that the magnitude and direction of the propulsion system thrust is a continuous function of time. The need to optimize such continuous thrust profiles presents new challenges in the area of trajectory design.

Trajectory optimization techniques currently in use need only model a series of discrete events: launch time, planetary flyby times (including characteristics of the flyby trajectory), plus any deep space maneuvers that may be required. This is by no means a trivial exercise when complicated multiple gravity-assist trajectories are considered, but the additional requirement of optimizing a continuous thrust profile severely strains, and in many cases exceeds, the capability of traditional techniques.

Current working methods for low-thrust trajectory optimization are calculus-of-variations-based. This formulation typically has difficulty with longer, more complicated trajectories and can expend large amounts of time executing inefficient searches. An on-going joint research venture being conducted between the Jet Propulsion Laboratory (JPL) and the University of Illinois at Urbana-Champaign's Computational Astrodynamics Research Laboratory (UIUC CARL) is investigating alternative methodologies in an attempt to discover new, more robust, and more efficient optimization algorithms. Automation of the optimization process using stochastic search techniques (e.g., simulated annealing and genetic algorithms) to drive the current optimization software is a major topic of research.

In this paper, a Pareto genetic algorithm is developed following existing methodologies[12] to perform a multi-objective optimization for low-thrust orbit transfers, expanding the previous work of Ref. 5. The optimization techniques discussed address the problem of low-thrust trajectory optimization, and provide for the generation of optimal sets or "families" of these trajectories. Generation of a trajectory family is accomplished by sorting the population according to an individual's Pareto optimality. Such sets illustrate the relationships between different flight characteristics, and provide compromise solutions when maximum performance (i.e. mass delivered to destination) is one of several requirements. Additionally, the new method provides increased robustness through its inherent separation of objectives and eliminates the objective conflict[13] which arises from classical techniques involving scalarization of multiple objectives, where improper assignment of weighting factors can result in bias towards certain objectives. The Pareto genetic algorithm is then combined with existing JPL trajectory optimization software in a hybridized manner to produce a more informed search, along with familiar and usable results.

1.2 Thesis Overview

The eight chapters contained within this thesis describe the construction of a hybridized Pareto genetic algorithm and its application to several interplanetary, low-thrust, spacecraft trajectory optimization problems. The first chapter provides introductory material for the reader and motivates the topic for investigation. Chapter 2 presents an introductory review of the concepts and operations involved in simulated annealing and genetic algorithms. Chapter 3 investigates characteristics of the types of the trajectory optimization problems that the algorithm is applied towards. Detailed search-space visualizations are performed to motivate the hybridization of a genetic algorithm with a localized search method. Various strategies for localized search are presented, and the method used in the hybrid algorithm identified. Chapter 4 describes the concepts of Pareto optimality and motivates the use of a genetic algorithm specifically designed for multi-objective optimization. Chapter 5 presents results verifying the performance of the Pareto algorithm constructed. Chapter 6 describes the hybridization of the Pareto genetic algorithm using a Baldwinian learning strategy. The resulting algorithm is then applied to three interplanetary trajectory optimization problems, with results included in Chapter 7. Chapter 8 provides conclusions and suggestions for further research. A list of references follows.

Chapter 2

STOCHASTIC OPTIMIZATION METHODS

Stochastic optimization techniques are optimization algorithms that utilize some random elements.[14] Unlike deterministic search algorithms that locate optima by systematically searching the solution space using gradient information, stochastic methods apply a degree of randomness to the decision-making process. The random nature of stochastic algorithms has discouraged their use in some circles, especially in the area of trajectory optimization where deterministic algorithms are favored;[15] however, the probabilistic elements in these algorithms provide them a capability not possessed by deterministic methods – the ability to hill-climb. Complicated trajectory optimization problems are often characterized by a highly multi-modal search space containing numerous local optima. In low-thrust interplanetary trajectory optimization this is certainly the case (see Section 3.4). Under these conditions, deterministic rules-based algorithms have difficulty while stochastic methods persevere.

In this study, the two most widely known stochastic techniques are investigated. Both of these methods are inspired by natural systems in physics and biology. In simulated annealing, the process by which atoms cool in molten metal to crystallize into their solid state is modeled.[16] Genetic algorithms implement the Darwinian concept of natural selection to evolve a randomly generated set of mediocre initial guesses into a population of optimal solutions.[17]

2.1 Simulated Annealing

Simulated annealing is a technique first developed by Scott Kirkpatrick et. al.[18] that simulates the cooling process in metals. At the heart of the simulation lies the Metropolis algorithm, which models the behavior of large systems of particles in equilibrium at a given temperature.[19] It is this component that is responsible for the stochastic nature of the algorithm and that gives the algorithm its probabilistic hill-climbing capability. The simulated annealing algorithm combines the Metropolis algorithm with a temperature schedule in an attempt to simulate the physical behavior of atoms as they cool from their liquid state into a minimal-energy solid.

2.1.1 The Simulated Annealing Metaphor: "Crystallizing" Solutions

The simulated annealing algorithm derives its name from the fact that it attempts to emulate the process of strengthening and hardening metals known as "annealing." The annealing process consists of heating a metal to a temperature sufficient to transform the metal to its liquid state, and then slowly cooling the metal at such a rate as to prevent brittleness in its hardened state. Heating of a metal to its molten state produces a substance whose atoms exist in a randomly aligned, high-energy state. The high energy level of the molten metal is due to a combination of increased kinetic and potential energies, which results in high temperature and changes in the material's physical structure. Each unaligned atom represents potential energy encased within the metal – the more closely packed the atoms, the lower the potential energy of the crystalline structure. Therefore at constant temperature, the energy level in the metal is dependent upon the number of misalignments. The objective of the annealing process therefore is not only to cool the metal, but to cool it in such a way that all of its atoms align in the same direction, thus achieving a perfectly ordered crystal and the lowest possible energy state.

As the metal cools, the total energy level in the amorphous substance decreases, atoms begin to align with one another to form crystals, and the metal begins to harden. Within these forming crystals, atoms are aligned with one another; however, their orientation may be different than that of a neighboring crystal. It is occasionally possible for one of these crystals to realign so as to match the alignment of its neighboring crystal and for the material to lower its overall energy level. Such realignments are a way for the metal to jump out of a locally optimal structure and proceed towards the globally optimal single-crystal structure.

Any realignment requires a temporary jump in the system's energy. The energy to realign is available as heat in the metal.[14] The temperature of the metal is a measure of the average kinetic energy of the atoms in that metal. Atoms within the metal therefore vibrate with an energy that depends on the temperature of the metal. As the temperature of the metal increases, more activity is present within the atomic structure. The probability of realignment is therefore temperature dependent, with realignments occurring quite often at high temperatures and only sporadically as the temperature approaches absolute zero. This temperature-dependent activity in the metal is the motivation behind the gradual cooling process. The metal's natural tendency is to obtain a low-energy state. Annealing theory portends that by cooling the metal very slowly, enough time will be spent at each temperature level for the metal to iron out its own imperfections through random realignments and the metal will eventually freeze in its perfectly aligned, lowest energy state.

Using the annealing process as a model, an algorithm metaphorical can be constructed that simulates the behavior seen in natural systems. The analogies that are drawn in constructing the optimization algorithm are given in Table 2.1. The crucial link in this simulation is the analogy drawn between the energy of a given configuration and the value of the objective function. As mentioned earlier, the annealing process attempts to derive a structural configuration of lowest energy. From the viewpoint of the simulated annealing algorithm, derivation of the problem parameter values that produce a minimal-cost solution is the objective.

Table 2.1 Simulated Annealing Analogies

Physical Annealing Process		Simulated Annealing
temperature	$\rightarrow$	simulated temperature
energy	$\rightarrow$	cost

atoms	$\rightarrow$	parameters
atomic structure	$\rightarrow$	parameter set
structural realignment	$\rightarrow$	perturbation of parameters

A more detailed description of the algorithm is shown below in Figure 2.1. The simulation of annealing is conducted by placing the Metropolis algorithm, a method for simulating atomic behavior at constant temperature (Sec. 2.1.2), on a gradually declining temperature schedule. The atomic structure, i.e. the set of problem parameters, is first initialized to a randomly determined state. The Metropolis algorithm is then run at the initial user-supplied simulated temperature for a number of iterations large enough to consider the system near thermal equilibrium. The system temperature is then decreased following a user-defined temperature schedule, and the Metropolis simulation is conducted for the new temperature. These temperature reductions continue until the simulated temperature of the system has reached absolute zero. Further discussion of the algorithm components can be found in Sections 2.1.2, 2.1.3, and 2.1.4.

```
Begin
  initialize()
  TemperatureSchedule: Do
    MetropolisAlgorithm: Do
      perturb(state i → state j)
      if energy(j) < energy(i) then
        update(i=j)
      else
        if exp[(energy(i)-energy(j))/T_k] > rand[0,1) then
          update(i=j)
        endif
      endif
    Until NEAR_THERMAL_EQUILIBRIUM
    decrementTemperature(T_k)
```

```
  Until FROZEN_SOLID
End
```

Figure 2.1 Pseudocode for the simulated annealing algorithm

2.1.2 The Metropolis Algorithm

Reportedly developed by its namesake, Nicholas Metropolis, during a Teller-hosted Los Alamos dinner party, the Metropolis algorithm is a technique used to simulate the behavior of large numbers of particles in thermal equilibrium at constant temperature. The Metropolis algorithm predicts the probability of a system of particles changing configuration to match a proposed state (e.g., the probability of a material's atomic structure realigning), and then updates the system state according to that probability. The probability for realignment can be expressed mathematically as,

$$\text{Probability of realignment} = e^{\frac{-\Delta E}{kT}} \tag{2.1}$$

with ΔE indicating the change in energy between the current and proposed states, T representing the temperature of the material, and k equaling Boltzmann's constant (1.3806 $\times 10^{-26}$ kJ/K). Thus the probability of the structure of the material changing from its current state to a new and different state is dependent upon the difference in energy between the current and potential state and the current temperature of the material.

It can be seen from Equation 2.1 that as the temperature of the system increases, the probability of accepting changes in the current state that raise the overall energy of the configuration also increase. This fact is illustrated in Fig. 2.2. Thus at high temperatures, structural realignments are highly probable even if they result in *inferior* configurations. Conversely, as temperature approaches absolute zero, the structure becomes increasingly immovable.

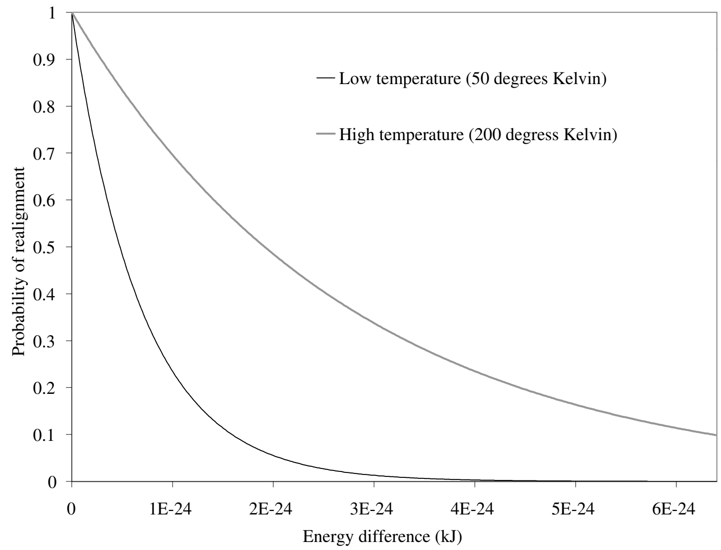

Figure 2.2 Probability of realignment as a function of temperature

The Metropolis algorithm simulates the behavior of systems of particles at constant temperature by taking an initial state, perturbing it in a random fashion, and probabilistically accepting a change from the previous state to the proposed state according to the Metropolis criteria. The Metropolis criteria for accepting a change in state are simply stated: given that perturbation of an existing state produces a proposed state, automatically accept this change in the state parameters if the proposed state is at a lower energy level (cost) than its predecessor. Conversely, if the proposed state has a higher cost – i.e., is a worse solution – accept the proposed state with a probability that is dependent upon the cost differential of the two states. This search method has been appropriately described as a "random biased walk" through the search space.[16] The search begins at a random point in the search space. It then proceeds to take steps through the search space that are generated in a random fashion, but that are also biased towards improving the solution.

The probability of acceptance is the mechanism that provides the hill-climbing capability of the simulated annealing algorithm. The possibility of producing a state that is worse than its predecessor in terms of its associated cost allows the algorithm to jump out of local minima. As the

Metropolis simulation proceeds over time at a constant temperature, the system has a chance to explore different configurations without becoming ensnared in a sub-optimal solution. As basins of attraction are located that provide lower and lower cost, the likelihood increases that the search will move to the lowest area on the cost surface due to the contingencies built-in to the Metropolis criteria. The deeper the basins of attraction become, the lower the probability of escape.

2.1.3 Temperature Schedules

The Metropolis algorithm only simulates system behavior at a constant temperature. To complete the annealing simulation, the Metropolis algorithm must be placed on a temperature schedule, with temperature gradually decreasing towards zero. The rate at which the system is cooled is typically controlled by user-defined parameters. Control is provided with regard to how long to run the Metropolis simulation at each temperature level, and how much to reduce the temperature each time it is updated. The user limits the number of states generated or accepted for each temperature level in the Metropolis simulation (assumed to be large enough for thermalization). After each Metropolis simulation, temperature is decreased from a very high initial amount by either subtracting a set scalar amount, or proportionally reducing its value.

How much time is allowed at each temperature, how high the initial temperature is set, and how fast the temperature is decreased are crucial elements to algorithm performance. The annealing process will produce a low energy ground state only if the starting temperature is sufficiently high and cooling is carried out sufficiently slowly, allowing time for thermal equilibrium to be reached at each temperature level[16]. A sufficiently high starting temperature is considered to be one at which steps through the search space are essentially random (i.e. no bias exists towards solution improvement). For this to be the case, the acceptance probability (Eq. 2.1) must be very close to or equal 1.0. The temperature level that produces this condition is generally calculable from Eq. 2.1 since a maximum ΔE (energy differential) can always be approximated.

The temperature of the system must also be decreased at a very slow rate if an optimal solution is to be found. If an initial temperature is set according to the guidelines above and the temperature is reduced as,

$$T_k = \frac{T_0}{\ln k} \qquad (2.2)$$

where T_0 represents the initial temperature, T_k represents the current temperature level, and k represents the number of temperature reductions, the simulated annealing algorithm is *statistically guaranteed* to find the optimal solution for any arbitrary problem.[20] This guarantee is one that no other optimization method can claim, thus increasing its appeal. There are, however, additional conditions that must also be in effect in order for this guarantee to hold true. Discussion of these conditions for convergence is left for Section 2.1.5.

The third parameter that defines a cooling schedule is the amount of time that the simulation is allowed to run at a constant temperature. Determination of this value is less formal than methods for calculating an initial temperature or temperature decrements. If the lowest ground state is to be found and the aforementioned guarantee is to hold, it is necessary that the system reach thermal equilibrium at each temperature. Determining when equilibrium has been reached can be done through standard methods of testing for thermalization, or through on-line measurements of energy variance at each temperature level during the annealing simulation. Measurements such as these can be used for determining the conditions that trigger temperature reduction, or for predicting the number of iterations to run the Metropolis simulation.

The methods for determining a cooling schedule described above are necessary for the statistical guarantee of convergence. These methods – and especially Eq. 2.2 – define a lower bound for the rate at which a system can be cooled using classical simulated annealing. Adherence to these rules, however, oftentimes results in an optimization algorithm that is too slow for practical use. A commonly used annealing schedule that deviates from the theoretically derived logarithmic one reduces temperature in an exponential fashion, calculating temperature as,

$$T_{k+1} = C \cdot T_k, \quad 0 < C < 1 \tag{2.3}$$

with T_k representing the current system temperature, T_{k+1} the updated temperature, and C a proportional reduction constant. Schedules such as these violate the lower bound on annealing rate and thus relinquish all claims to guaranteed convergence; however, they possess the attractive feature of greatly decreasing the runtime of the algorithm. Overly rapid cooling of the system – known as simulated *quenching* – is a strategy that is oftentimes successful in locating optimal or near-optimal solutions. Quenching is a compromise strategy that can produce reasonably good results in shorter amounts of time than simulated annealing. Due to faster runtimes and the general acceptability of near-optimal solutions to difficult optimization problems, simulated quenching has become a widely used technique.

2.1.4 Perturbing the State

The third essential operator for a simulated annealing algorithm is a state generation procedure. This is the procedure that produces the proposed state changes and defines how the algorithm steps through the search space. Although perturbation of the state in the simulated annealing algorithm is a necessary process, the algorithm does not include rules specifying *how* to perturb the state. The Metropolis algorithm states only that proposed changes to the system should be accepted with a probability that is dependent upon the change in energy. This leaves somewhat open the issue of how to generate proposed states.

There still remains, of course, the matter of being able to meet the statistical guarantee for convergence. Not all state generation functions will meet this guarantee. In order to have a valid procedure, the state generation function must meet two criteria: 1) it should be *ergodic*, and 2) it should not violate *detailed balance*.[20] These are complicated sounding conditions, but they have relatively straightforward definitions. In order for the state generation procedure to be ergodic, every state generated must be attainable from every other state in a finite number of iterations. In other words, there must not exist any barriers or conditions within the state generating procedure that make it less likely to search some areas of the search space than others in a finite time simulation. In order for the procedure to not violate detailed balance, it cannot be biased in any way with regard to how the transition is made from one state to the next. This condition is most clearly described mathematically,

$$W(A \to B)P(A,t) = W(B \to A)P(B,t) \tag{2.4}$$

with $P(A,t)$ representing the probability of being in configuration A at time t, $W(A \to B)$ the transition probability of going from A to B, etc. In other words, given the probabilities of being in states A or B at a given time, the probability of transitioning from state A to state B must be equal to the probability of transitioning from B to A.

Both of these criteria thus involve issues of non-bias and accessibility. Although simulated annealing is a "random biased walk" through the search space, the bias must not be introduced in the state generation function. The bias referred to in such a description is instead the bias introduced by the Metropolis algorithm that favors transitions to lower-energy states. In order to have an appropriate state generation procedure that meets the guarantee of convergence, all points in the search space must be equally attainable with no one-way transitions or bias of any kind.

State generation methods employing random number generation are typically used to efficiently meet these criteria.

In classical simulated annealing algorithms, the state generation procedure implements a random number generator that produces numbers based on the probability density given by the well-known Gaussian or normal distribution. There are, however, various ways of producing random numbers and thus various ways of sampling the search space. Any state generation procedure that is ergodic and does not violate detailed balance is valid and will provide the statistical guarantee of convergence. Altering the state generation procedure does require development of appropriate acceptance criteria (Eq. 2.1) so as to provide compatible probability distributions, but it can also result in derivation of a lower theoretical bound on the temperature schedule following the proof outlined in Ref. 20. Alternative state generation procedures have been investigated and have produced techniques such as Fast Simulated Annealing (FSA)[21] and Adaptive Simulated Annealing (ASA) which in its earliest form was known as Very Fast Simulated Re-annealing (VFSR).[22] The former implements a Cauchy distribution for state generation and acceptance criteria, while the latter uses a generating function designed to allow an exponential cooling schedule. For further details on these advanced methods see Refs. 21 and 22.

2.1.5 Conditions for Convergence

As has been noted, simulated annealing is the only optimization algorithm that can claim to statistically guarantee an optimal solution. It has also been noted that this claim is not unconditional. There are several criteria that a simulated annealing algorithm's operators must satisfy before this guarantee can be met. These conditions for convergence include:

1. Initial temperature must be set high enough so as to allow random sampling of the search space when the simulation begins.

2. The rate of temperature reduction must not violate the lower bound on the temperature schedule (defined by the state generation procedure and acceptance criteria).
3. Thermal equilibrium must be reached at each constant temperature level before reducing the temperature to the next level.
4. The state generation procedure must be ergodic and must not violate detailed balance.

Upon closer inspection, it can be seen that the simulated annealing algorithm's claims for guaranteed convergence are not all that profound. The convergence criteria basically require that, in order to guarantee an optimal solution, the algorithm must be given enough time to adequately search the solution space. By exaggerating this requirement some, it is apparent how such a guarantee can be made. As the time allotted for the search approaches infinity, the search becomes exhaustive. Therefore as computational resources increase, it becomes easier to meet the temperature schedule conditions for arbitrary problems. Computation resources are of course not infinite. For difficult optimization problems requiring long run-times, a compromise between algorithm speed and the quality of the solution is frequently made. As previously discussed, this usually comes in some form of simulated quenching which claims no guarantees.

A much more difficult issue to resolve involves the requirement of ergodicity. While it is mathematically possible to derive a state generation mechanism that satisfies detailed balance, it is difficult to prove that the procedure is ergodic. This is due to the fact that ergodicity is a problem-dependent characteristic. As a practical matter, ergodicity describes the problem search space. A state generation procedure may not be inherently biased toward any given area of the solution space, but if the problem search space is structured in such a way that it limits the access of the search mechanism to a particular subspace, that mechanism is essentially non-ergodic. Well-behaved problems with continuous search spaces, void of noise or other aberrations are considered ergodic. Non-ergodic search spaces arise in difficult optimization problems when large energy barriers (e.g. large non-continuous changes in the cost surface, large plateaus of constant cost providing no search information, or very noisy surface areas) divide the search space up into multiple facets. Features such as these either eliminate the prospect of ergodicity or reduce the state generation procedure to one that is *quasi-ergodic* – a procedure theoretically able to attain all possible states, but with a high degree of improbability given a finite amount of time. Executing multiple runs starting from different points in the search space and comparing final results is one way to test for ergodicity. If all control parameters have been set to comply with the given conditions of convergence and the solutions obtained reside in different regions of the search space, it is likely that the search space is non-ergodic. Testing for ergodicity can also be accomplished through direct inspection of the search space, as was done in Section 3.4.

As can be seen from the previous discussion, the conditions which must be in place for the simulated annealing algorithm to ensure an optimal solution greatly constrains the number of problems for which convergence can be guaranteed. While convergence of the algorithm to an optimal solution can statistically be guaranteed in the theoretical realm, application of the algorithm to real-world problems brings to light the impracticality of many of the algorithm's criteria for convergence. Although claims of guaranteed convergence might be somewhat inflated, simulated annealing remains a powerful optimization algorithm for multi-modal problems due to the probabilistic hill-climbing capabilities it possesses. Simulated annealing is thus not a panacea, but rather a powerful alternative in the family of stochastic optimization algorithms.

2.2 Genetic Algorithms

A genetic algorithm is a stochastic optimization technique based upon the concept of natural selection in Darwinian genetics.[23] John Holland and his graduate students at the University of Michigan originally developed the first simple genetic algorithm (SGA) in 1975.[24] The structure of this algorithm differs substantially from simulated annealing. Rather than working with a single solution consisting of the problem variables as in simulated annealing, a genetic algorithm maintains a pool of solutions. Each solution consists of a set of problem parameters, typically encoded to form a string of binary numbers. The algorithm attempts to manipulate this genetic pool of information through application of the genetic operators *selection*, *crossover*, and *mutation* in an attempt to "evolve" an optimal solution.

2.2.1 The Genetic Metaphor: A Method for "Natural Selection"

Charles Darwin introduced his theory of natural selection in his 1859 work, "On the Origin of the Species."[25] The process of natural selection that Darwin described is the mechanism by which evolution takes place. Natural selection is often described as "survival of the fittest," a term commonly attributed to Darwin but actually coined and popularized by the 19th century English naturalist and philosopher Herbert Spencer.[26] The principle maintains that organisms best suited to survive in their environment are more likely to reproduce and pass their genetic material on to the next generation. Those organisms with less advantageous traits are less likely to survive long enough to reproduce.

A genetic algorithm is an attempt to mimic this mechanism albeit in a very limited way. Although genetic algorithms use the ideas and language of genetics, they do not begin to approach the level of complexity in a living organism's reproductive system. Genetic algorithms are instead a simplification of these natural processes. Evolution of a population of individuals *is* simulated, but individuals in this population are not dynamic organisms in any way. No real-time interaction between an individual and their environment or other individuals is simulated.[*] Individuals simply represent static points in the solution space.[†] Taking these static solutions, evaluating how well they

---

[*] An exception to this is the case of hybridized algorithms that attempt to improve individuals through the use of localized search techniques. In keeping with the natural selection metaphor, localized improvement strategies have been termed "learning" strategies – the organism interacts with its environment and learns to improve its fitness during the course of the evolutionary simulation. See Section 3.5.
[†] This again may or may not be the case when learning strategies are involved.

solve the problem, and slightly altering or combining parts of solutions in attempts to improve them simulates the process of evolution.

Individuals represent problem solutions by each maintaining a set of parameters that can be substituted into a merit function of some kind. In a simple genetic algorithm these parameters are encoded in binary format and connected together to form binary strings in an attempt to emulate a chromosome of a living organism. Transforming the parameter set for a given solution in such a way facilitates the application of the genetic operators. Other techniques for representing solutions exist, including alternate encoding methods or maintenance of real-valued parameters;[23] however, a binary coding strategy is straightforward and thus is commonly used.

$$F(x,y) = x^3 + y^3 - 3x - 12y$$

$$x = 3 \Rightarrow 00011$$
$$y = 5 \Rightarrow 00101$$
$$\Downarrow$$
$$\mathbf{0001100101}$$

$$F(0001100101) = 83$$

Figure 2.3 The encoding, decoding, and fitness evaluation of an individual

Storing problem parameters in binary format requires that encoding and decoding subroutines be included components of any SGA. Individuals are constructed using these encoding and decoding routines in the manner represented in Fig. 2.3 above. In order to construct an individual representing a problem solution, real values of problem parameters are first encoded into binary format. A value of 3, which the genetic algorithm has produced for the x variable in the example above, encodes to its binary form 00011. A y value of 5 encodes to 00101 in binary. The binary numbers are then concatenated into a string of bits representing a single solution, in this case

0001100101. In order to obtain a figure of merit for this particular solution, the encoding process is reversed and an individual's parameter set is decoded for objective function evaluation. The user must construct this objective function which determines how the system is to be optimized. This objective function is typically a mathematical function of the real values of problem parameters, which produces a scalar fitness value. In the example depicted above, the objective function is a function of the variables x and y given by, $F(x,y)=x^3+y^3-3x-12y$. The decoding process restores the parameters x=3, y=5 which, when substituted into the objective function, yield a value of 83. The binary string 001100101 thus possesses a figure of merit equal to 83.

Since the values included in the example above are integer values, they can be represented exactly in binary format. In many optimizations, of course, not all problem variables take on integer values. A common strategy for dealing with this problem is to store each parameter's bounds in memory, and then discretize each variable over its respective range. The user establishes an upper and lower bound and a resolution for a real variable. The range for the variable is then divided into a discrete number of values dictated by the given resolution. An integer representing the real-valued parameter's position in this discretized range is then used as the binary representation. Thus in most genetic algorithms, not all real values are represented for real-valued variables. It is this integer-valued parameter which is translated to and from its binary form and that is manipulated by the genetic operators. The upper and lower bounds for a particular variable are relied upon during the encoding or decoding of a variable to obtain the real value of the variable. As the resolution parameter becomes very large for a given variable, that variable can be modeled more precisely. Increasing a variable's precision, however, also means increasing the length of the binary string that represents it. Therefore it is not always advisable to set the resolution parameter such that machine precision is maintained for floating point variables since this will lead to substantial string lengths and extensive computation.

In the language of genetics, the binary string of bits that result from the encoding process is referred to as a *chromosome*. Each chromosome contains the encoded problem parameters, or *genes*, which in turn are composed of individual bits, or *nucleotides*, of information. The term *chromosome* is used interchangeably with *individual* since an individual possesses only a single chromosome. Furthermore, an individual's set of binary-valued parameters is alternately known as its *genotype*, while the set of real-valued parameters is considered its *phenotype*. The encoding and decoding routines then convert an individual's phenotype to its genotype and vice versa. It is an individual's phenotype that is used to evaluate its objective function or *fitness*. A listing of the terminology commonly used in genetic algorithms is included in Table 2.2.

Table 2.2 Analogous terminology in genetics and genetic algorithms

Terminology in Genetics		Genetic Algorithm
chromosome	↔	binary string
gene	↔	binary parameter
nucleotide	↔	bit
locus	↔	string position
allele	↔	decoded parameter value
genotype	↔	set of encoded parameters
phenotype	↔	set of decoded parameters
fitness	↔	objective function value

Once the parameter set has been encoded and concatenated to form a binary string, it is in a form that the genetic operators can access and manipulate. Exactly how these operators work is described in Sections 2.2.2, 2.2.3, and 2.2.4. The framework of the overall algorithm is given below in Fig. 2.4. The algorithm begins by initializing the population to a set of randomly generated solutions. The random sample of the solution space is then evaluated using the objective function constructed for the problem, and each member of the population is assigned a fitness value. The algorithm then enters the main generational cycle. This loop iteratively applies the genetic algorithm's operators to the population until a pre-established maximum number of generations (determined by the user) is reached. Individuals are first selected into the mating pool based upon their evaluated fitness and the selection scheme implemented. Fit individuals are selected randomly for crossover where information is exchanged between solutions. The mutation operator is then applied to the resulting new solution. All solutions generated by the reproduction cycle are stored in a temporary population. At the end of the cycle, the population is updated and evaluated, and the process of applying the genetic operators is repeated once again. Once the maximum number of generations has been reached, the final population is output.

```
Begin
  initialize(population)
  evaluate(population)
  GenerationalCycle: Do
    selection(population → mating pool)
    ReproductionCycle: Do
      crossover(parent1, parent2 → newChromosome)
      mutation(newChromosome → child)
    Until NEW_POPULATION_FULL
    updatePopulation(newPopulation → population)
    evaluate(population)
  Until MAXIMUM_GENERATIONS
  outputFinalPopulation(population)
End
```

Figure 2.4 Pseudo-code for a simple genetic algorithm

2.2.2 Genetic Operators

A simple genetic algorithm possesses three basic functions that operate on a population of solutions. These three genetic operators are *selection*, *crossover*, and *mutation*. The selection operator places individuals in the mating pool based upon their evaluated fitness. Crossover randomly selects two "parent" individuals from the selected group, and mates them to produce one or more "child" solutions. The mutation operator scans through each child individual's chromosome and probabilistically changes, or mutates, the genetic information encoded within.

Within each category of genetic operator, a myriad of advanced techniques is available. Since documenting all such methods is not the purpose of this thesis, only the operators used in this study are described in the following sections. References to more advanced techniques are included where appropriate.

2.2.2.1 Selection

The selection operator, as its name implies, is the operator that enforces the principle of natural selection. This is the operator that determines which individuals will have the opportunity to pass on their genetic material. After a population has been generated and evaluated, the selection operator extracts the fittest individuals from the population for reproduction. Like most components of a genetic algorithm, this selection process contains randomized elements so that the selection process is not *completely* dependent upon fitness. Selection of individuals with the highest fitness values is favored, but their selection is not guaranteed. The probabilistic elements involved allow for the possibility of inferior solutions passing on their genetic information as well so that a degree of diversity is maintained. Some degree of diversity throughout the search is required so that the solution space may be adequately sampled.

The probabilistic elements of selection are best described by example. Figure 2.5 below illustrates a selection method known as *tournament* selection. In this case *binary* tournament selection is illustrated, as it was the one of the selection operators incorporated into the genetic algorithm used in this thesis. In this selection scheme, two individuals are randomly pulled from the population and compared against one another in terms of fitness. The individual with the highest fitness is placed in the mating pool, while the other is discarded. As can be seen from this simple example, highly fit individuals are not guaranteed selection. Higher fitness values only increase the probability that an individual will produce offspring. This is the case not just for tournament selection, but all selection operators.

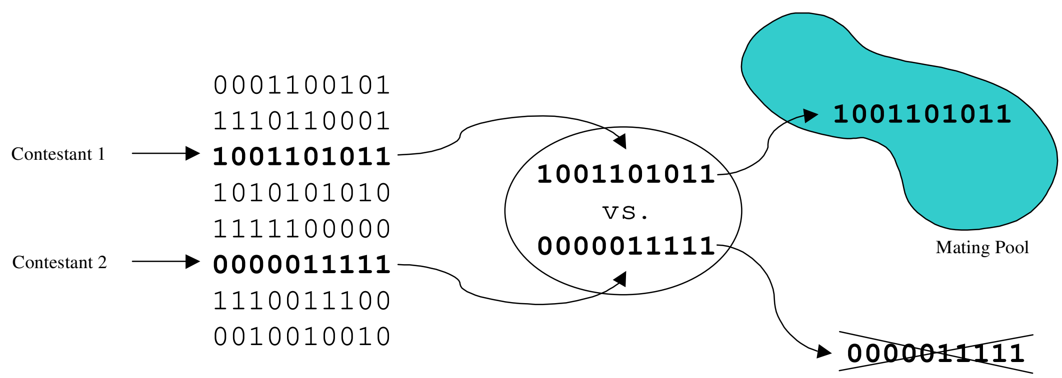

Figure 2.5 Two individuals chosen in binary tournament selection

An alternative to tournament selection is *proportionate* selection. Selection schemes such as these calculate the probability that an individual will be selected by comparing its fitness to the population at large. The probability of selection p of an individual i with fitness f_i is given by,

$$p_i = \frac{f_i}{\sum_{j=1}^{k} f_j} \tag{2.5}$$

where k is the size of the population. All members' fitness values are thus summed, and an individual's probability of selection is determined by dividing its fitness value by this sum. These probabilities can be viewed as pieces of a pie or sized slots on a roulette wheel (Fig. 2.6). This was the approach taken in the original form of selection known as "roulette-wheel" selection. [27] In order to select an individual, the wheel is spun. Whomever the marker is pointing to when the wheel comes to rest is the individual selected.

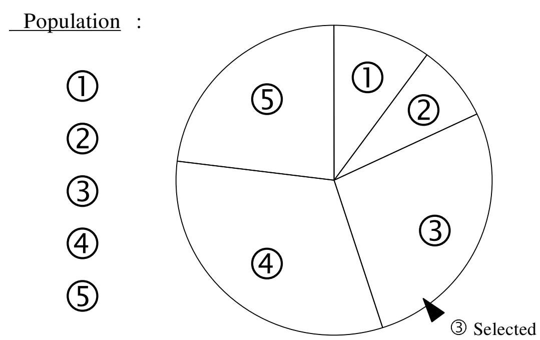

Figure 2.6 Roulette-wheel selection

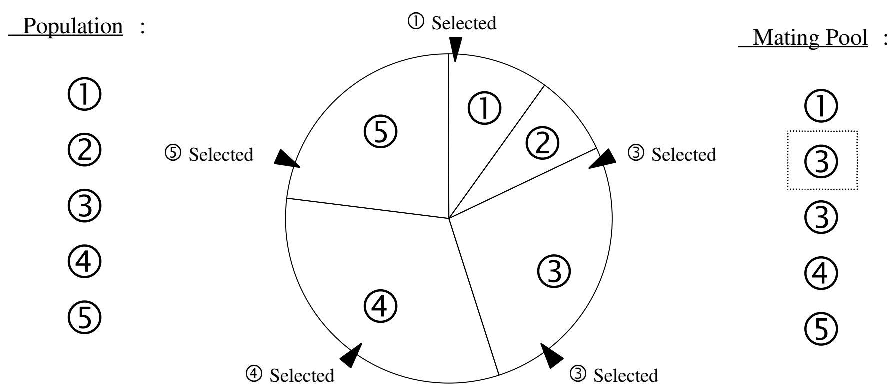

Figure 2.7 The stochastic universal selection procedure

The Pareto genetic algorithm used in this study and discussed later in Sec. 4.3 uses the proportionate selection scheme known as stochastic universal selection (SUS). Tournament selection is unavailable due to the fact that the Pareto GA incorporates a niching operator (see Sec. 2.2.4). The pairing of these two operators results in instabilities in the population, thus an alternative selection scheme must be chosen.[28] The SUS scheme is illustrated above in Fig. 2.7. The probabilities of selection for all individuals are calculated according to Eq. 2.5, but instead of establishing a single marker and requiring a number of spins, *k,* to fill the mating pool, *k* markers are evenly distributed about the roulette wheel and one spin is required. The mating pool is then filled with the individuals to whom the distributed markers point. A selection scheme such as this is much less computationally intensive while remaining probabilistically equivalent to other proportionate schemes such as roulette wheel selection.[29]

There are many other selection schemes besides the tournament and proportionate methods listed here. Even within these categories, a number of different strategies exist which may be executed to produce different algorithm behavior. A thorough analytic comparison between the most common selection schemes can be found in Ref. 29. As to the question of which selection method is best, a general answer remains elusive.

2.2.2.2 Crossover

A genetic algorithm's crossover operator is its primary search mechanism and the one that separates it from the broader superset of evolutionary search algorithms. The crossover operator is the reproductive mechanism of the GA. After selection has been imposed on a population of solutions and a mating pool has been generated, crossover is applied. In all cases presented in this thesis, *single-point* crossover is used. This operator works by first extracting two randomly selected solutions from the gene pool and probabilistically determining whether or not crossover will take place. The probability of crossover is a user-defined parameter. If crossover is to be performed, a *single point* on the chromosomes is randomly located, the chromosomes are spliced at this point, and genetic information is swapped between the two.

The process is illustrated in Fig. 2.8. The crossover point is always common to both chromosomes and is generated using a uniform random number generator so that there is no bias when splicing. In this example, crossover takes place between the 5th and 6th nucleotides of both chromosomes. The binary strings after this point are swapped resulting in two new chromosomes. The new chromosomes must then have the mutation operator applied before they are copied into the new population.

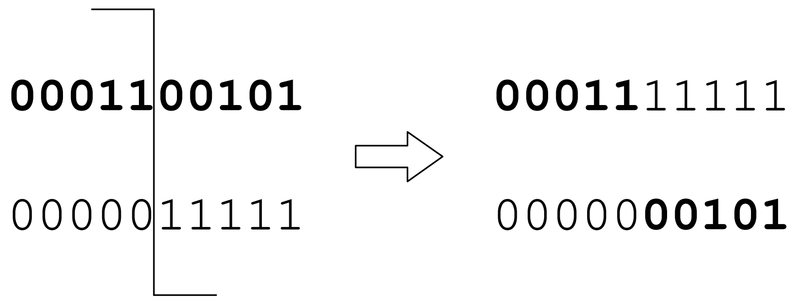

Figure 2.8 Single -point crossover

Like the selection operator described earlier, a multitude of crossover strategies exist. It is fairly straightforward to modify the classic crossover algorithm to generate very different modes of behavior. Among these simple modifications are multiple -point or uniform crossover where more than one crossover points are determined.[23] It is also easy to incorporate control structures which create either one or two children, accept only the better of the two children, compare children to parents and accept the best pair, accept only the best two individuals out of both pairs, etc. Crossover operators such as these and their resulting performance are described in Refs. 30, 31. More sophisticated crossover operators such as those that must be implemented when alternative coding mechanisms or real -codings are used are described in Refs. 32, 33.

2.2.2.3 Mu tation

The mutation operator is a lesser operator in the philosophy of genetic algorithms, but an important one nonetheless. This operator serves to restore some of the genetic information that may be lost after application of selection and crossover. Th e selection and crossover operations purposefully weed out inferior individuals. These weaker individuals, however, may include in them beneficial traits that are lost when the individuals are discarded. Mutation is thus applied to each individual in the population after selection and crossover have taken place in an attempt to restore some of this lost information. This application involves sequentially stepping through each bit in an individual's chromosome and probabilistically flipping it (changing a 1 to a 0 or vice versa) according to a pre-defined probability of mutation. The probability of mutation is a user-defined parameter and, due to its secondary nature, is usually very small. Figure 2.9 below illustrates mutation operating on a chromosome of length 10. As the mutation operator steps through each bit position, a uniform random number between 0 and 1 is generated. If this number matches or falls below the probability of mutation, the current bit is flipped to its alternate value. In the case illustrated in Fig. 2.9, the mutation operator has stepped through to the seventh bit of the chromosome before mutation occurs.

$$000111\boxed{\mathbf{1}}111$$

$$000111\boxed{\mathbf{0}}111$$

Figure 2.9 The mutation operator

As with the previous two genetic operators, variations on simple mutation abound; however in this study, only simple mutation (as described above) is implemented. Details of alternate mutation operators are not presented here, but can be found in Ref. 23. Although mutation is a secondary operator in genetic algorithms, it is the primary search mechanism in other evolutionary search methods. This explains the multitude of alternative mutation operators. Classic evolutionary algorithms combine selection with mutation and contain no crossover routine. In effect, the mutation operator randomly perturbs the parameter set producing a random walk through the search space. This is an important point to note, as this description is not dissimilar to the state generation procedures in simulated annealing algorithms.

2.2.3 Setting Genetic Algorithm Control Parameters

As was the case with simulated annealing algorithms, assignment of control parameter values for genetic algorithms is oftentimes not a straightforward task. The complex stochastic processes that occur in GAs make control parameter optimization difficult at best. Further complicating matters are the changes in GA behavior that result from permutations of genetic operator strategies (e.g., tournament selection with multiple-point crossover and simple mutation vs. stochastic universal selection with single-point crossover and Fletcher-Reeves mutation,[23] etc.). Despite the complex nature of the analysis and the variety of operator combinations, an abundance of informative material can be found regarding the determination of control parameters. The usefulness of this material, however, depends upon the level of complexity of the optimization problem and the commonality of the genetic operators implemented. Nonetheless, these resources detail important issues surrounding the determination of the GA control parameters, mutation rate, crossover rate, and population size, and provide information as to how to attempt to size these parameters appropriately.

Setting the probability of mutation for a simple mutation operator is perhaps the most straightforward procedure. (This, however, may be due to less rigorous study of the effect of mutation in genetic algorithms due to the secondary nature of the operator.) It has been found in empirical genetic algorithm studies that in order to obtain good results, the frequency of mutation should be on the order of one mutation per thousand bit transfers.[23] In practice, this information is often translated to produce a probability of mutation inversely proportional to the population size. As is the case in natural populations, this is a very small number and thus the lesser import of the operator.

Establishing the probability of crossover can be a much more complicated task. Early work in function optimization by De Jong establishes a crossover probability of 0.6 as being a good compromise between what he termed *on-line performance*, an averaged value of all function evaluations, and *off-line performance*, an averaged value of the best performance values.[27] Other studies since have established higher probabilities of crossover (p_c=1.0) when accurate selection procedures that produce low stochastic sampling errors are in place.[34] Even more recent studies have examined the effect of recombination on the problem of adequate mixing in GA populations.[30,35] All of these references should be studied for a thorough understanding of the issues surrounding the crossover operator, as there does not exist any generalized equation for crossover rate determination.

Population sizing is perhaps the least well-defined area in GA parameter determination. Empirical studies have demonstrated that the size of the GA population is a major factor in determining the convergence quality of GAs, with large populations resulting in better solutions.[36] References 34, 37, and 38 take further steps in an attempt to quantify exactly how big a population must be to ensure good solutions, and provide a great deal of insight and helpful rules for sizing populations. These studies, however, are limited to simple genetic algorithms applied to tailored optimization problems where the search space characteristics are known. Complex engineering optimization problems where little is known about system behavior are not examined. Practically speaking, the population sizing equations provided have limited utility. This is especially true when modifications are made to simple genetic algorithms to incorporate advanced operators or include problem-specific behavior. As complexity increases and modifications are introduced, fewer of the assumptions from basic research studies hold true, and the methods derived become less reliable. Population sizing for difficult problems thus commonly lapses back into the realm of empirical determination.

Control parameter identification is therefore not the definitive process one might hope for. As is the case with simulated annealing, an abundance of analytical work has been conducted in attempts to derive explicit equations for the setting of control parameters. Practical considerations, however, often force those implementing GAs to move away from theoretical limits, instead designating control parameters based on guidelines from research but adapted for specific problem behavior.

2.2.4 Niching and Other Advanced Operators

In addition to the standard genetic operators of selection, crossover, and mutation, many additional operators have been developed to enhance the capabilities of genetic algorithms. Reference 23 contains descriptions of many of these advanced techniques, which are unfortunately beyond the scope of this work. Two particular advanced operators, however, are made use of in this study and thus are worthy of further description. The primary advanced technique utilized in this thesis involves modifying the fitness evaluation procedures in order to produce a multi-objective optimization algorithm. This technique incorporates the concept of Pareto optimality, one of the fundamental topics of this thesis. Since this is an area of primary interest, it is discussed in detail in Chapter 4. The second of these advanced techniques is known as *niching* or *fitness sharing*. Niching has become a commonly used advanced operator that serves to diversify the population of solutions in an attempt to prevent premature convergence.

The intent of niching is to establish sub-populations of solutions at each local optimum in a multi-modal search space. In many complex optimization problems, the search space contains multiple peaks (if maximizing the objective function) where the value of the objective function nears optimality to differing degrees. A genetic algorithm without the niching operator typically handles such a problem by attempting to locate the peak of greatest magnitude, gradually migrating all solutions in the population to this optimum. This behavior may be what one hopes for – as long as the optima converged upon is in fact the global optimum or near to it. It is often the case that for difficult problems with multi-modal search spaces the GA will have difficulty locating the global optimum, and many times will prematurely converge to a sub-optimal solution.

In such situations, a niching strategy may be employed in order to maintain diversity in the population for longer periods of time. A more diverse set of solutions provides more genetic material for exchange through crossover, thus allowing for a more extensive search. In addition, the global optimum may not be the only feasible solution to the optimization problem. Each local optimum may represent a potentially useful near-optimal solution, with some trade-off existing between the objective function value and another solution characteristic. Thus the generation of stable sub-populations is often advantageous for more than one reason.

This formation of stable sub-populations is a phenomenon that also exists in nature. The development of niching operators is yet another instance of genetic algorithmists drawing upon examples in nature to create analogous GA constructs. When nature is presented with an environment where multiple resources of varying quality exist, stable sub-populations of organisms surrounding these separate niches are formed by forcing similar individuals to share available resources.[39] This natural exploitation of separate resources manifests itself in the form of speciation. Through speciation, different species develop. Each species possesses a common set of characteristics that enable it to take advantage of a particular type of available resource. Whatever task a species member specializes in places it in a niche within its habitat. A niche may therefore be viewed as an organism's job or role in the environment.[23]

The niching scheme used in this study is based upon the concept of fitness sharing. This concept is described in Ref. 23 using the example of a two-armed bandit or slot machine. Imagine a two-armed slot machine as illustrated in Fig. 2.10. Each arm has associated with it a different payoff: the left arm a jackpot of $25, and the right arm $75. The expected payoffs of these two arms are initially unknown. Further suppose that a population of 100 players exists, and that each player receives the full payoff amount for whatever arm they line up behind. It is clear that once the players ascertain the discrepancy in payoff amounts, all players will quickly move to line up behind the right arm to obtain the larger jackpot. This is what happens in a GA without niching when a local optimum is located. Population members quickly move to the currently known highest payoff without adequately searching for even better payoffs.

Now suppose that the situation is altered slightly so that with each jackpot, the payoff received must be divided up and shared amongst all the individuals in a particular line. If all the individuals playing were to line up behind the right arm, the resulting payoff for each individual would be $75/100 = $0.75. If only one of those individuals had broken ranks to line up behind the left arm though, the payoff for that individual would have been a substantially larger sum of $25. Thus motivation for balancing the number of individuals in each line has been produced through the introduction of sharing. In this example, this balance occurs when 75 players line up behind the right arm, and 25 line up behind the left arm. Such a proportional distribution results in the equal payoff to each individual of $75/75 = $25/25 = $1.

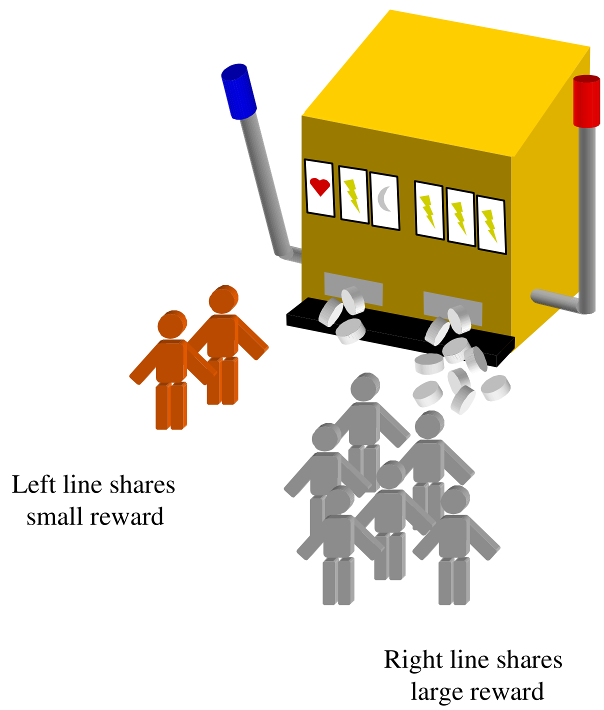

Figure 2.10 Sharing amongst individuals in two lines for the two-armed slot machine

In the slot machine example above, the respective queues of individuals clearly define the group that the payoff is to be shared amongst. Such divisions do not exist in a GA's population of solutions.

These niches have to be created. Subdivision of the GA population is accomplished using what is known as a *sharing function* (Fig. 2.11).[40] This function defines the radius of the neighborhood around an individual in which fitness sharing will occur, and the degree to which sharing will be applied. The sharing function is given by,

$$Sh(d) = \begin{cases} 1-\left(\dfrac{d}{\sigma_{share}}\right)^{\alpha}, & \text{if } d < \sigma_{share}; \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

where the sharing function, $Sh(d)$, is a function of the distance-metric, d, between two individuals in a population, σ_{share} is the radius of the sharing neighborhood, and α is a power-law parameter which defines the rate at which the degree of sharing decreases with proximity. Either an individual's phenotype or genotype can be used to calculate the required distance metric. In this study, a phenotypic sharing scheme is used resulting in a distance metric equivalent to the distance between solutions in the decoded parameter space.

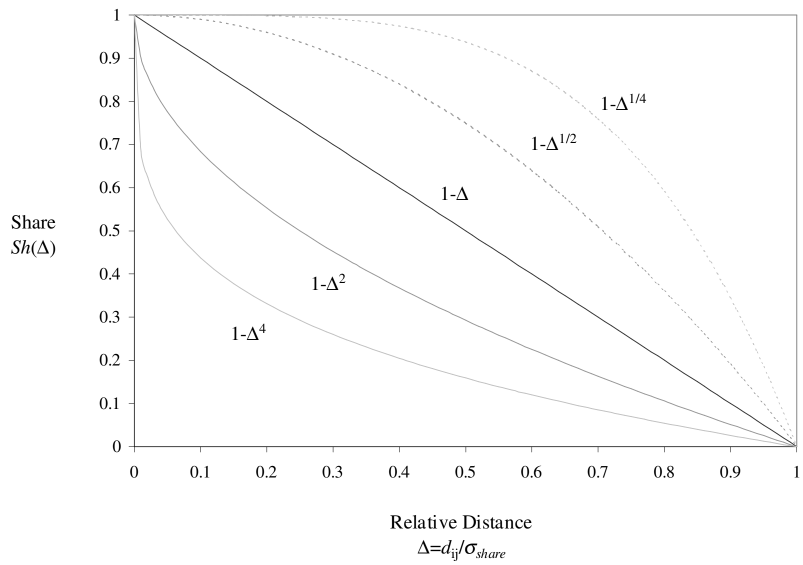

Figure 2.11 Power law sharing functions

In the niching procedure, this sharing function is used to penalize solutions according to their degree of similarity to others in the population. The population is examined on an individual-by-individual basis to determine the number of other members that reside within the neighborhood of the current examinee. The size of this neighborhood is determined by the niching control parameter, σ_{share}. A penalty value is associated with each individual residing within this neighborhood. The magnitude of the penalty value falls off as proximity to the examinee decreases by the power α. As can be seen from Fig. 2.11, a value of α=1 corresponds to a linear reduction in magnitude with the effective penalty approaching zero as individual proximity approaches the neighborhood boundary, σ_{share}. Such a linear sharing function is termed a "triangular sharing function."[23] The penalties corresponding to the individuals residing within this defined neighborhood are tallied, and the current examinee's fitness is penalized by an amount proportional to that number at the end of each generational cycle.

This derated fitness value is calculated using the equation,[23]

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} Sh(d(x_i, x_j))} \tag{2.7}$$

where $f_s(x_i)$ is the adjusted fitness of individual i, $f(x_i)$ is the original fitness of individual, and n is the population size. With the sharing function defined above, the minimum value of the summation will be 1.0, due to the fact that when $i=j$, $Sh(d_{ij})=1.0$. In such a case, the only individual within the current individual's neighborhood is itself, and fitness should not be reduced.

The use of a niching operator introduces the additional control parameters α and σ_{share}, which must be assigned values by the user before an optimization can take place. The value for α is typically set to 1.0 so as to produce a triangular sharing function. The user can tweak this value to produce subtle differences if desired. No explicit method for setting α exists. The effects of setting α to values greater or less than 1.0 can be seen in Fig. 2.11.

To determine the σ_{share} parameter in p-dimensional parameter space, the assumption is made that q niches exist, each in a hypersphere of radius σ_{share} such that each sphere encloses 1/q of the volume of the space.[39] The σ_{share} parameter can then be calculated by,

$$\sigma_{share} = \frac{\sqrt{\sum_{k=1}^{p} \left(x_{k.\max} - x_{k,\min}\right)^2}}{2\sqrt[p]{q}} \tag{2.8}$$

where $x_{k,\max}$ and $x_{k,\min}$ are the respective upper and lower bounds for each problem variable in the p-dimensional parameter space. For an optimization problem with a single parameter (p=1), the above equation reduces to,

$$\sigma_{share} = \frac{x_{\max} - x_{\min}}{2q} \quad (2.9)$$

For a complete derivation of these equations and further description of both phenotypic and genotypic sharing schemes, see Ref. 39.

These equations are very useful; however, as is the case with many other existing rules for parameter sizing, they require prior knowledge of the search space. The number of niches, q, is information not generally known before an attempt at optimization is made. This potential pitfall is usually sidestepped by setting q to a desired number of niches. Several iterations of the same optimization problem using various values for q can also be executed and compared to determine a well-suited value of q experimentally.

Niching is therefore a technique that is straightforward in conceptualization and implementation. Explicit rules aid the user in determining values for the niching control parameters. Like the other genetic operators, however, these rules are limited. Nonetheless, niching is a powerful mechanism that is useful in many optimization problems.

Chapter 3

HYBRIDIZATION OF OPTIMIZATION ALGORITHMS

Having provided a basic description of the stochastic optimization algorithms used in this thesis, more specific detail with regard to how they were employed can now be provided. As was mentioned previously, stochastic optimization algorithms require an objective function for optimization. This objective function may be as simple as an algebraic function of one variable, or as complex as a call to an external program that returns data to the algorithm. It is the latter method that is implemented in this thesis. The algorithms developed rely on a Calculus-Of-Variations (COV) optimizer – the Solar Electric Propulsion Trajectory Optimization Program (SEPTOP) – to obtain measures of constraint satisfaction and spacecraft performance for the various trajectory problems to which they are applied. Each stochastic method is used as a driver for the SEPTOP software, essentially automating its usage by acting as a kind of "smart" user. The following sections describe the integration of the COV optimizer with the stochastic optimization methods, and the reasoning leading up to the eventual development of hybridized methods where SEPTOP is used as a localized search procedure.

3.1 SEPTOP: Calculus of Variations Optimization Software

SEPTOP is a preliminary mission planning tool developed by NASA's Jet Propulsion Laboratory that uses a two-body, Sun-centered, low-thrust solar-electric propulsion model.[41] Classical Calculus-Of-Variations is used to obtain a maximum final mass for a spacecraft whose performance parameters are supplied by the user. Specifics of the desired trajectory such as origin, destination, and trajectory type (fly-by, rendezvous, multiple fly-by, etc.) are also specified. This results in a two-point-boundary value problem (TPBVP). Along with the spacecraft and trajectory parameters, the user provides initial estimates for costates (Lagrange multipliers). The state and costate differential equations are then forward integrated. Terminal constraints on the states and costates that result from the COV formulation must be satisfied. An optimal trajectory as defined by SEPTOP is thus one that meets the constraints produced by the TPBVP while maximizing final spacecraft mass.

The convergence of SEPTOP to an optimal trajectory can be highly dependent on the user's initial guess and the relative difficulty of the mission. As the number of intermediate planetary flybys and total number of revolutions about the sun increases, the user's initial guess for the trajectory costates must move closer to the converged values for the TPBVP to be successfully solved. This artifact gives some indication of a multi-modal search space. In addition, as trajectory complexity increases, sensitivity to the initial guess increases, with small increments to the costates producing erratic changes in the degree of convergence. This ill conditioning indicates the presence of noise and very small basins of attraction for existing optima. As a result, determination of complex trajectories with SEPTOP becomes more art than science, and requires much time and effort. Development of an algorithm that is able to eliminate much of the tedium associated with complex trajectory generation using SEPTOP is thus motivated.

3.2 Integration of SEPTOP and Stochastic Methods

The interface between SEPTOP and the stochastic methods used in this study is straightforward and independent of stochastic optimizer type. Since both the genetic and simulated annealing algorithms require only a scalar objective function value for a given problem and no gradient information, it is simple to construct a SEPTOP calling routine compatible with both methods. SEPTOP reads in information for formulating a trajectory optimization from an input file. A SEPTOP input file consists of two parts: problem-specific information for a given trajectory optimization problem, and a set of costates used to generate a trajectory. The problem-specific information is supplied by the user. This file defines spacecraft parameters, initial and final spacecraft positions, and any other problem parameters or constants. The initial guess for the costates is then appended to this file. Under normal operation, the user would supply the initial guess for the Lagrange multipliers. When using a GA or SA to drive the process, these costates are supplied by the stochastic method. Within the GA or SA algorithm, bounds for each of the costates are supplied, and the algorithm produces sets of costates within these bounds through the use of its operators. Instead of evaluating an objective function included within the GA or SA's main program, a call is made to an external subroutine that concatenates the spacecraft and problem parameters with the generated costates, and uses the resulting input file to execute SEPTOP. SEPTOP is then executed in a non-search mode which does not make use of its internal gradient-based search methods, but instead simply returns a normalized measure of convergence for the trajectory produced from the given set of costates. SEPTOP thus provides a direct mapping between the costate values and a convergence error value. This convergence measure can then be used as the objective function value by the stochastic optimizer.[‡]

In essence, the stochastic method replaces the human user and automates the optimization process. Instead of a user iteratively guessing the initial costate values, executing SEPTOP, and adjusting the costates to achieve better results, this process is automated through the use of a GA or SA. Just as a human user might proceed, each stochastic method guesses a solution, receives feedback as to the quality of that solution, and guesses a new solution for evaluation based on what it has learned from past trials. The optimization ceases when good results have been found.

[‡] As one might imagine, additional information can be returned from SEPTOP to further qualify a solution's merit. Furthermore, control over any of the problem parameters can also be yielded to the stochastic optimizer if desired. Descriptions of these strategies can be found in Chapter 6.

3.3 Search-Space Visualization

Initial optimizations implementing the above formulation produced varying degrees of success. For the most part, however, the same obstacles that commonly occur with human users also arose in the automated process. Convergence to an optimal trajectory was highly sensitive to the initial guess of the Lagrange multipliers. This sensitivity suggested a highly multi-modal search space with an abundance of noise and needle-in-a-haystack (NIAH) type optima. Such a search space would be highly problematic, especially in the case of the simulated annealing algorithm which requires ergodicity for convergence. Initial trials indeed suggested a non-ergodic search space, as optimizations starting from different initial random locations converged to different areas of the search space.

Due to this behavior, visualizations of the search space were attempted. The high dimensionality of the problem, however, limited such attempts to partial visualizations. Visual information was generated using a known solution to a particular problem, fixing all but a few of the problem variables and varying the remaining free variables over their respective ranges. The results were then plotted to obtain visual estimations of the ergodicity of the search space.

3.3.1 1-Dimensional Visualization Technique

Search space visualization in problems with low dimensionality is a rather straightforward process. For optimization problems with only one variable, varying the single problem parameter and recording the objective function value produced at each value generates a 2-dimensional curve. The optimization process then reduces to finding the lowest point on this curve, assuming that the task is to minimize the cost function. For problems with two independent variables, a 3-dimensional cost surface can be generated by representing objective function values as altitudes corresponding to points in the search space. The optimization process is then likened to finding the lowest point on the cost surface. Optimization problems possessing more than two variables, however, are much more difficult to visualize. Due to limitations in human perception, it becomes difficult to depict objects in any higher-dimensional space.

One way to visualize problems of high dimensionality is to examine them in some reduced form. Keeping this in mind, a technique was developed to aid in the visualization of search spaces of high dimensionality by simply looking at them one dimension at a time. A 1-dimensional visualization, as it is termed in this thesis, fixes all but one of the problem variables to some baseline value. In the 1-D visualizations illustrated in Figs. 3.1 and 3.2, the baseline value is an optimal solution – optimal in the sense defined by SEPTOP. The free variable is then varied over its respective range to generate an array of objective function values. These objective function values are then plotted over the independent variable's range in order to obtain visual information about the behavior of the search space. The process can be thought of as viewing a cross-section of the multi-dimensional search space, with each graph presenting information on how difficult it is to approach the optimal solution from that dimension.

Figure 3.1 illustrates 1-D visualizations of three of the nine Lagrange multipliers for the optimization problem of an Earth-Mars flyby trajectory. It can be seen from these plots that the problem appears relatively well behaved. Although exhibiting some multi-modal features, the search space is not so treacherous as to deter the use of either stochastic method. These three costates are representative of the remaining six costates for the problem. If all trajectory optimization problems using the SEPTOP software were this benign, there would be no need to apply new techniques. This problem, however, is a simple one for which a human user in a short amount of time can easily obtain an optimal solution. It is unfortunately not representative of the difficult kinds of problems we would like to be able to solve.

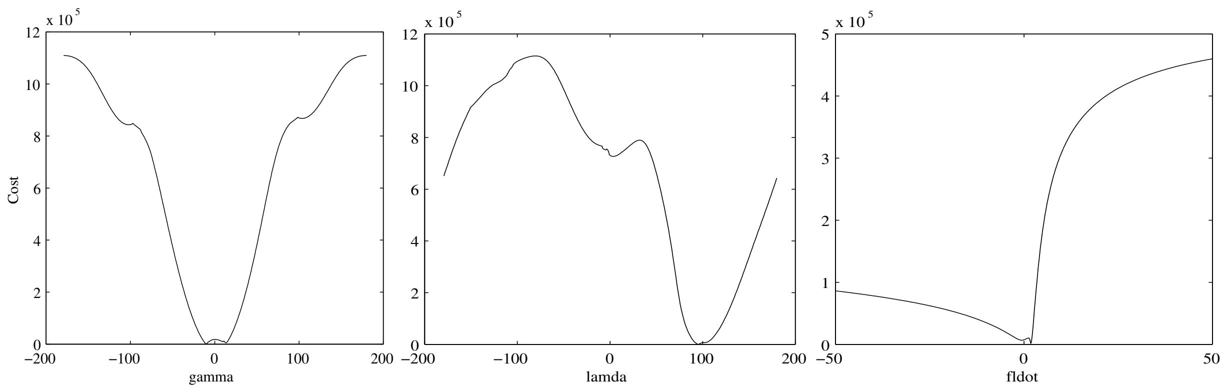

Figure 3.1 Earth-Mars flyby 1-D search space visualization

A second visualization is that of an Earth-Mercury rendezvous trajectory executing several revolutions about the sun during its approach to Mercury. As can be seen from Fig. 3.2, this is a much more difficult problem and one that is not readily solved by a human user. Optimal solutions are much harder to locate due to their NIAH nature. The existence of noise and large energy barriers further increase problem complexity, making optimization all but impossible for gradient-search methods unless initiated very near a local optimum.

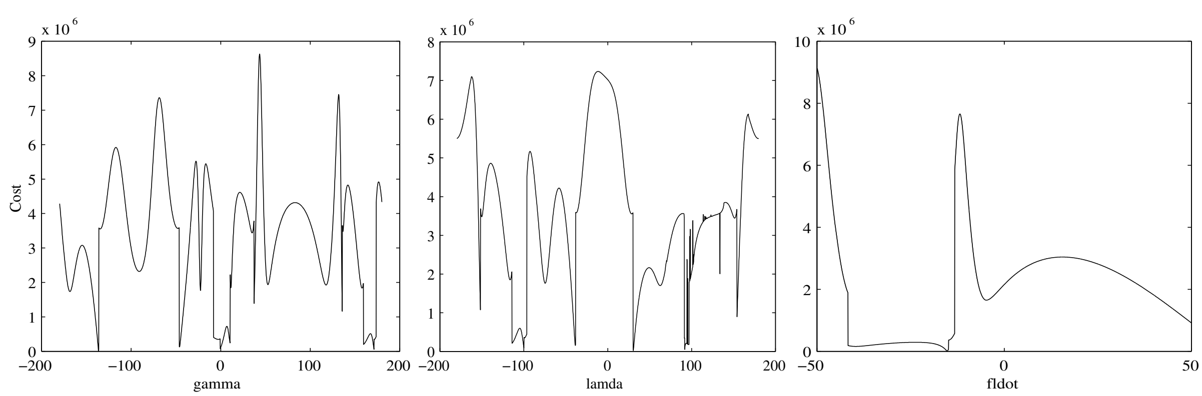

Figure 3.2 Earth-Mercury rendezvous 1-D search space visualization

3.3.2 2-, and 3 -Dimensional Visualization Techniques

In order to obtain a somewhat broader view of the search space, the technique developed in Section 3.3.1 was expanded to generate what amounts to a multi-dimensional cross section of the search space. Instead of holding all but one of the problem variables constant and using the free variable to produce an array of objective function values, two of the problem variables are freed to produce a matrix of objective function values. For each point in this 2-dimensional space, a cost function value is determined. From this information, a cost surface is generated to provide further information about the problem search space. As was mentioned earlier, the objective of the optimization process then becomes finding the lowest point on the surface.

This same process can be carried one step further and a third variable freed to produce a visualization of three problem dimensions. The search space is represented in 4 dimensions – 3 spatial dimensions (two problem variables plus their associated cost) and one temporal. This is simply done by generating a series cost surfaces, each with a different value for the third variable. The first two problem variables act as latitude and longitude as before, with cost represented as altitude. The third variable acts as a time variable, animating the cost surface over its range of values. The result is a sequence of images like those illustrated in Figs. 3.3 and 3.4.

Figure 3.3 illustrates the same Earth-Mars flyby problem examined above. The added dimensionality reveals that this problem is less simplistic than the previous 1-D analysis suggested; however, it is still a well-behaved problem with smooth surfaces that shift gradually as the temporal variable is varied. This is in stark contrast to the cost surfaces produced by the Earth-Mercury rendezvous problem shown in Fig. 3.4. The large energy barriers, NIAH optima, and noisy regions manifest themselves in three dimensions as tall ridges and spiky barriers surrounding isolated optima. As the third variable is change over the sequence of frames, the surface shifts dramatically. Ridges shift outward as elevated regions collapse. The effect is similar to a boiling surface.

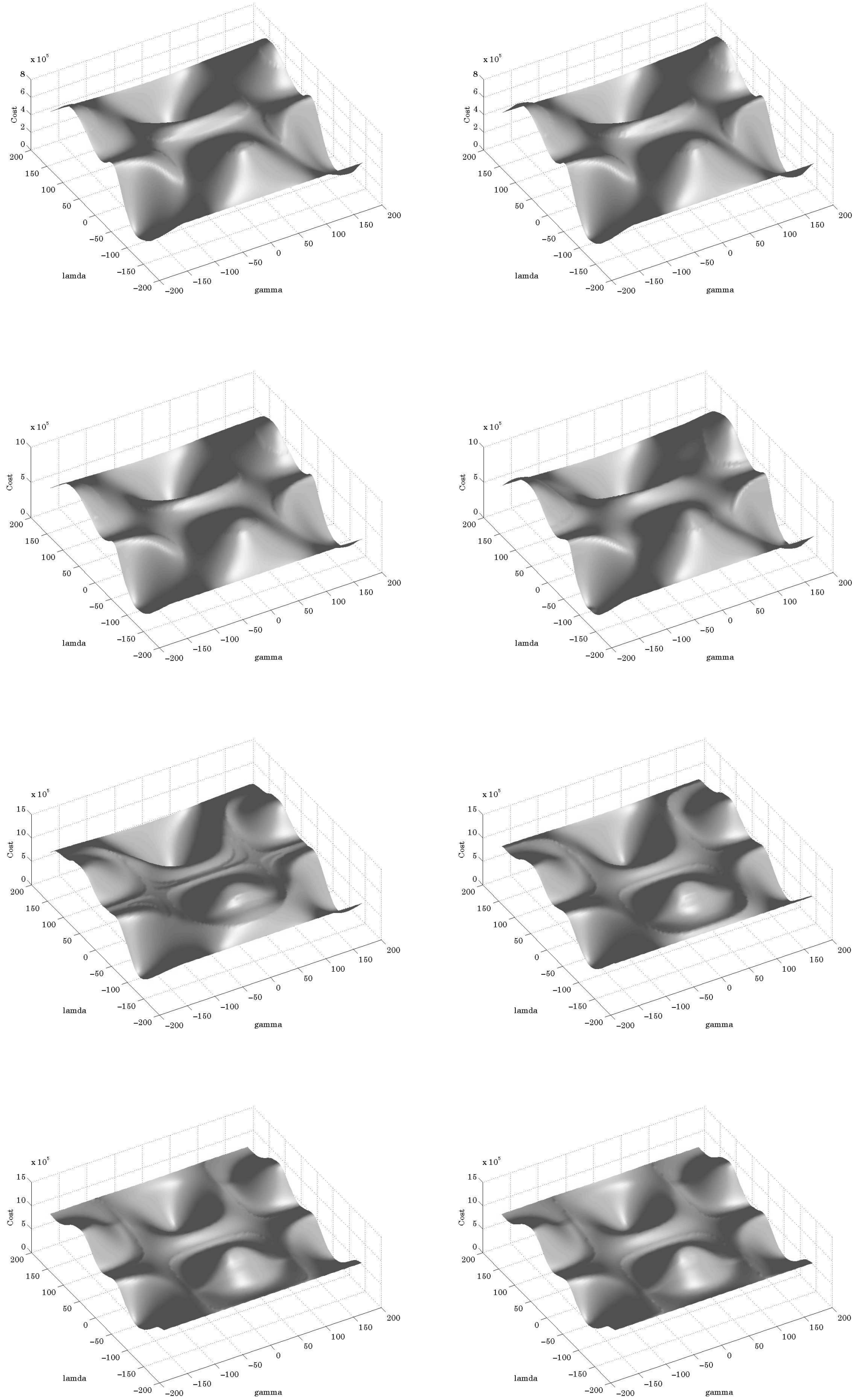

Figure 3.3 Earth -Mars flyby 3 -D search space visualization

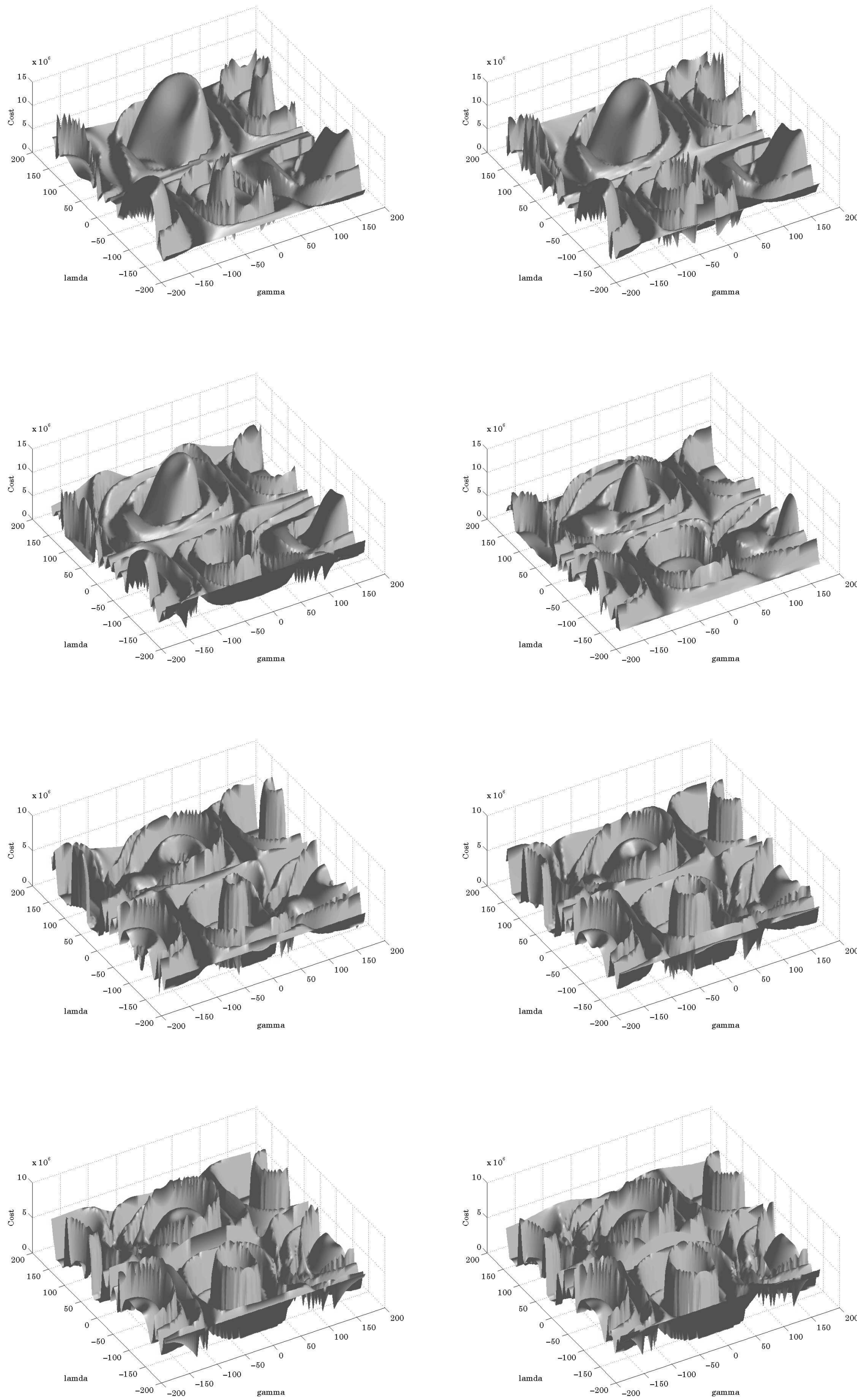

Figure 3.4 Earth-Mars rendezvous 3-D search space visualization

3.3.3 Implications of Analysis

The preceding search space analysis reveals the complex nature of the costate search space. The partitioning of the search space into facets bounded by large energy barriers discourages the use of simulated annealing. Difficult problems fail to meet the SA criteria of ergodicity. The barriers that separate areas of the search space hinder the operation of the SA algorithm's search mechanism and eliminate any guarantees of convergence. As a result, the remainder of this thesis focuses on the genetic algorithm approach. This method is more likely to search a larger area of the search space given the fact that it takes what amounts to a "shotgun" approach to optimization. Instead of maintaining a single solution and attempting to walk that point through the search space as the simulated annealing algorithm does, the GA starts by sprinkling many solutions over a large portion of the search space. The GA is therefore able to automatically bypass many of the existing energy barriers. A similar effect may be accomplished by constructing a parallel simulated annealing algorithm and starting the algorithm at multiple points; however, such an algorithm's operations, it can be argued, approach that of a genetic algorithm with an advanced crossover operator. Therefore only a genetic algorithm is studied in this thesis, and the topic of parallel simulated annealing is left as an area for further research.

3.4 Localized Search Techniques

Given the nature of the search spaces produced by difficult problems, there is a low probability that either stochastic method would be able to locate optimal solutions consistently. The NIAH nature of the optima makes their discovery more a result of chance than efficient search. Such problems make apparent the need for additional procedures to aid in locating these optima. One such procedure is to incorporate a localized search method into the stochastic optimizer. In most instances, a localized search algorithm has to be constructed using some form of gradient-based search. In the case of the algorithm on hand, localized search is incorporated simply by "turning on" the SEPTOP optimizer and taking advantage of its built-in search operations. The incorporation of localized search is discussed in the following sections.

3.4.1 The "Learning" Metaphor

Genetic algorithms are very efficient at identifying the region within the search space where an optimal solution occurs, but usually do no possess the precision needed to pinpoint its exact location.[42] To provide this precision, a localized search procedure or "learning strategy" is oftentimes implemented. Learning strategies are incorporated as a means of fine-tuning solutions produced by the GA. GA solutions are used as the starting point for a localized search procedure that attempts to improve the solution even further, usually by implementing some form of deterministic search. This formulation produces a hybridized search method that alternates between stochastic and deterministic search during the course of the optimization.

Local improvement methods can be analogized as the learning process population members undergo to improve their fitness during their lifetimes. Organisms succeed not only through natural selection on a generation-by-generation scale, but also by learning about and adapting to their environment during their lifetimes. Localized search methods simulate this learning process. Figure 3.5 below illustrates the effect of a learning strategy. Localized search transforms the search space, widening areas where feasible solutions exist in order to attract more solutions. This expansion of existing basins of attraction can be very helpful in identifying NIAH optima. Without localized search, identification of such optima is hit-or-miss. With localized search, any solution in the region surrounding and optimum has the potential to discover that optimal solution by searching its surroundings to improve its fitness.

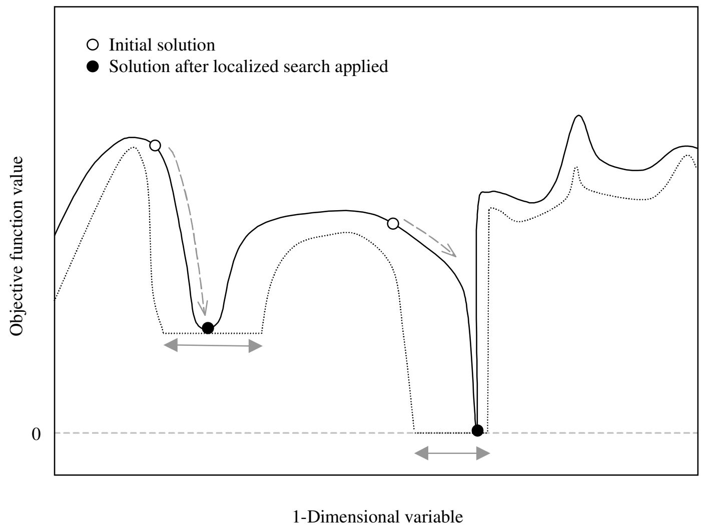

Figure 3.5 The effect of localized search

As can be seen from the example above, localized search can be especially effective when the problem search space is highly multi-modal. Learning strategies are also useful in constrained problems. Since GA's have no built-in function for handling problem constraints, appended penalty functions or other modifications to the objective function must be made in order to enforce constraints. Localized search is one effective way of enforcing constraints and, in essence, "repairing" chromosomes so that they conform to problem restrictions and produce feasible solutions.[43,44] Since the GA's objective function in this study is in part a measure of constraint violation returned by SEPTOP, localized search can be viewed as such a correcting mechanism.

More than one method exists for implementing localized search. The following sections detail two learning strategies. The first is a method that may seem effective but in fact leads to premature convergence. The second provides a more correct implementation.

3.4.2 A Lamarckian Learning Strategy

Learning strategies, by their nature, take an existing solution and attempt to make improvements through additional search. As a result, solutions before learning possess different features than after. These differences take the form of: 1) an improved fitness; and 2) a new parameter set that yields the improved fitness. Various options therefore exist with regard to how an individual is updated after learning is applied. Either an individual's fitness, their parameter set, or both may be updated as a result of the local improvement.

Since reproduction of population members in a GA is based upon fitness, it would seem that an update of fitness should take place. Updating an individual's parameter set (phenotype[§] and genotype) only is readily dismissed. Such an update would lend no advantage to the individual since its measure of optimality – and thus its potential to procreate – as seen by the GA would remain unchanged. It would also seem advantageous to update an individual's parameter set, therefore passing on beneficial genetic information obtained from the search. A logical choice would then seem to be to update both the individual's fitness and parameter set.

Adopting such a strategy, however, has unintended consequences. Updating an individual's parameter set in this manner is equivalent to the "inheritance of acquired traits," a discredited theory of heredity proposed by Jean-Baptiste Lamarck in the early 19th century.[45] This strategy is therefore often referred to as a *Lamarckian* learning strategy. In essence, Lamarck's theory states that an organism's genetic code can be altered during its lifetime through adaptation to its environment. These adaptations can then be passed down to future generations.

Implementation of this flawed theory usually manifests itself as premature convergence in genetic optimization. Replacing the existing genetic information in population members with the improved genetic information obtained from localized search eliminates much of the diversity in the population. As mentioned in Chapter 2, maintaining diversity is the key to solving difficult problems for genetic algorithms. Using a Lamarckian learning strategy, all individuals in the vicinity of a

---

[§] Some confusion exists in the literature with regard to the discrepancy between phenotype and fitness. Here we regard phenotype as an individual's decoded genotype, not its fitness value.

particular local optimum are lumped together and given the same genetic material (Fig. 3.6). This quickly eliminates population diversity and leads to premature convergence of the algorithm.

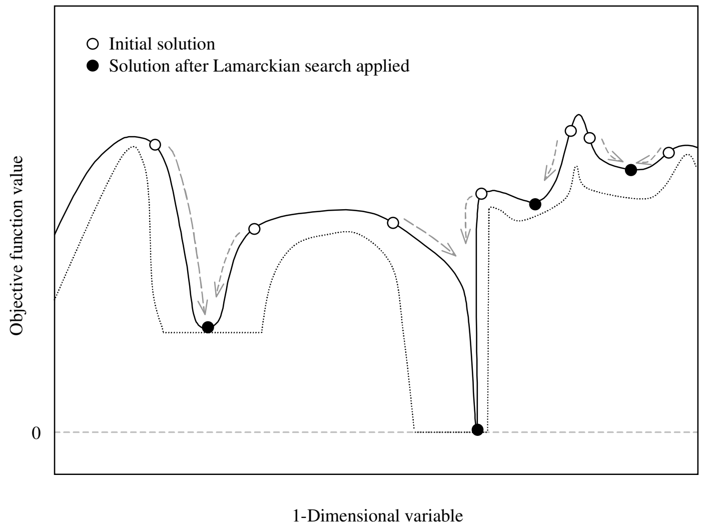

Figure 3.6 The effects of a Lamarckian learning strategy

3.4.3 A Baldwinian Learning Strategy

An alternate and more correct strategy for incorporating localized search is to exercise the remaining option of those discussed in the previous section and update an individual's fitness without affecting its genotype. Such a strategy is often termed a *Baldwinian* learning strategy after James Mark Baldwin, a late 19th century psychologist who postulated that individual learning could explain evolutionary phenomena that appear to require Lamarckian inheritance of acquired characteristics.[46] Baldwin's theory describes how individual learning can guide evolution. In terms of optimization, learning has the ability to smooth the fitness landscape thus facilitating the optimization process. This effect is often referred to as the Baldwin effect.

Baldwinian search is often more effective than Lamarckian due to that fact that diversity can be maintained within the population while also incorporating the benefits of local search.[47] The updating of an individual's fitness has the effect of recording a potential fitness for a given set of problem parameters (Fig. 3.7). These potential fitness values can then be used to guide the evolutionary process while still retaining a diverse pool of genetic information with which to conduct the search.

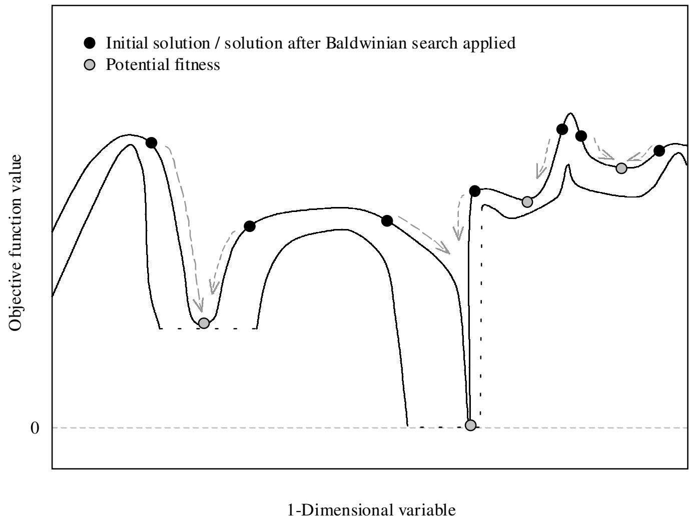

Figure 3.7 The effect of a Baldwinian learning strategy

Of the two methods, Baldwinian learning is preferred and is the method used in this thesis. Without the Baldwin effect, the genetic algorithm used in this study had limited success in locating obscured optima. It is worthy to note, however, that several investigations have been made into the use of partial Lamarckianism.[43,44,48] In these learning strategies, Lamarckian evolution is relied upon some small percentage of the time, with the dominant learning strategy being Baldwinian. This combination has in some instances proven to be as successful as Baldwinian learning alone, and has the added benefit of reducing search time due to gradual movement of the GA population to optimal regions of the search space by the Lamarckian operators. Lamarckian evolution is therefore not completely lacking utility. Application of partial Lamarckianism possesses potential as an area for future research.

Chapter 4

PARETO OPTIMIZATION

The genetic algorithm described thus far is a powerful tool, but despite the addition of a Baldwinian learning strategy, some shortcomings still remain. In particular, the GA in its current form utilizes SEPTOP to derive optimal solutions using the SEPTOP convergence error as the sole measure of fitness for population members. As was mentioned earlier, SEPTOP attempts to drive this error term to zero while maximizing the final spacecraft mass. While these two objectives are desirable, they can lead to trajectories that the hybrid algorithm considers optimal,[**] but which would be automatically excluded from consideration by a mission planner. Examples of such solutions – solutions often encountered in initial trials of the hybrid GA – would be trajectories that require an excessive time of flight, approach the sun too closely, or possess other characteristics judged inappropriate for the mission at hand.

The solution to this problem is to provide the algorithm with additional criteria with which to qualify "good" trajectories. While final spacecraft mass is the typical measure of performance, it is not the only desirable trait in trajectory optimization. Minimizing time of flight is quite often an equally attractive mission objective, and one that is inversely related to maximizing deliverable mass. Other objectives may also arise in the mission planning process that are deemed equally advantageous. Combining these objectives forms a multi-objective optimization problem and can aid the optimization algorithm in finding trajectories that are considered truly "good" solutions. The following sections detail a robust method for multi-objective optimization that not only allows the GA to produce more informed solutions, but provides the additional capability of generating sets of optimal solutions representing the trade-offs that exist between multiple, competing objectives.

4.1 Multi -Obje ctive Optimization

Multi -objective optimization, as its name implies, differs from single -objective optimization in that the intent is to optimize a system with more than one objective. As in single -objective optimization, the objective(s) may have any nu mber of equality or inequality constraints imposed upon them. This can be represented mathematically as, [49]

$$
\begin{array}{lll}
\text{Minimize/Maximize:} & f_i(x) & i = 1,2,\ldots,N \\
\text{Subject to:} & g_j(x) \leq 0 & j = 1,2,\ldots,F \\
 & h_k(x) = 0 & k = 1,2,\ldots K
\end{array}
\qquad (4.1)
$$

Rather than searching for the solution which yields the globally maximal (or minimal) value for a single objectiv e, the "best" solution is found by simultaneously optimizing several objectives. These types of optimization problems have traditionally been solved by associating to each objective a weighting factor, then combining the weighted objectives into a single scalar objective. Such reduction techniques eliminate the need for a more complex multi -objective algorithm, but introduce new parameters in the form of the weighting factors. [13] Individual solutions are highly sensitive to the magnitudes of these weighting factors. The user must become familiar with the exact relationship between objectives in order to determine the proper weighting values that will yield the desired result. Determination of the correct weighting factors can, in fact, become an optimization process of its own.

In addition, the result of the single -objective optimization will be a single solution rather than a set of compromise solutions illustrating the relationship between different objectives. This is acceptable for cases in which optimality of all objectives coincide in the same solution, but in most cases where adjustments beneficial to one objective are detrimental to others, such a technique provides only a

---

[**] Ignorance of traditional measures of trajectory "goodness " is often advantageous. The absence of

single point on what may be an expansive "front" of possible solutions. In the case of genetic algorithms in particular, such a formulation fails to take advantage of the population-based nature of the technique. It therefore becomes desirable to develop a more robust multi-objective algorithm, capable of identifying the trades between objectives and able to make better use of the population-based GA to produce sets of optimal solutions.

4.2 Pareto Optimality

Pareto optimization, named after Italian sociologist and economist Vilfredo Pareto (1848-1923), is the principle of optimizing multiple competing objectives. The problem is essentially one of finding optimal solutions based on criteria that have inverse relationships. The concept is succinctly described as follows,[50]

> *A Pareto optimal solution is not unique, but is a member of a set of such points which are considered equally good in terms of the vector objective. This space may be viewed as a space of compromise solutions in which each objective could be improved upon, but if it were, it would be improved at the expense of at least one other objective.*

The intent then is to derive a set of elite solutions from a larger population of candidate solutions through simultaneous comparison of several criteria. This may not seem a straightforward task; however, a simple methodology has been developed[23] for systematically sorting solutions for Pareto optimality. The procedure is described in the following sections.

---

pre-conceived notions regarding optimality oftentimes results in the production of novel trajectories.

4.2.1 Non-dominance vs. Dominance

The search for Pareto optimality has thus far been framed as the simultaneous optimization of several objectives. Viewed as such, the problem may seem rather convoluted. An alternative conceptualization would be to think of a solution as Pareto optimal if, for a given set of objectives, no other solution exists which is superior to that solution in *all* objectives. In other words, a Pareto optimal solution is one where no other solution exists which categorically dominates it. Such a solution thus is termed "non-dominated." If a solution exists which is superior in *some* objectives, then that solution may also constitute a point on the curve of Pareto optimal solutions (depending upon how it compares to the rest of the candidate solutions), but it does not disqualify the original solution from being Pareto optimal.

Similarly, a solution is only considered to be non-Pareto optimal if it is categorically dominated by another solution in the candidate pool. If a solution is worse than any other solution being considered under all criteria, then there is no possibility that the solution is a member of the set of Pareto optimal solutions. Such solutions are termed "dominated." When considering dominated solutions, it should also be noted that different levels of dominance are possible. A dominated solution will always be categorically inferior to some non-dominated solution in the set of Pareto optimal solutions; however, a dominated solution may be dominated by another intermediate dominated solution as well, who's objective vector may place it somewhere between the two in the hierarchy of solutions. These levels of dominance provide a means of fully characterizing a group of solutions by separating them into Pareto optimal "fronts," with each front possessing a different level of Pareto optimality. The utility of this aspect of the paradigm may be somewhat difficult to grasp, but becomes apparent when implemented from within the Pareto genetic algorithm.

Using the concepts of non-dominance and dominance, Pareto analysis can be simplified to a search for non-dominance. The search consists of sorting candidate solutions into groups of non-dominated and dominated solutions, with non-dominated being the favored trait. The non-dominated solutions discovered by the search constitute the set of compromise solutions to the multi-objective optimization problem. In this way, Pareto optimization can be seen as the dual to classical optimization where global dominance is the desired attribute.

In order to illustrate the concepts of dominance and non-dominance further, a simple example is presented: the problem of minimizing both coordinates for a set of points, {(0,5),(1,3),(2,4)}. In this set, point 1 is inferior to points 2 and 3 in its second coordinate; however, point 1 is superior to points 2 and 3 with respect to the first coordinate. Given then that point 1 is not *categorically* dominated by any other point in the set, point 1 is said to be non-dominated. Point 2 is dominated neither by point 1 nor point 2 since it is superior to these two with regard to its second coordinate. Point 2 is thus also non-dominated. Point 3, while superior to point 1 in its second coordinate, is dominated with respect to *both* coordinates by point 2, therefore it is a dominated individual and not Pareto optimal.

4.2.2 Pareto Optimal Fronts: An Example Population

In order to illustrate the existence of levels of dominance and the concept of Pareto fronts or "ranks" in a population of potential solutions, a more sophisticated example is presented. Suppose, for example, that a mission designer is attempting to develop a mission to Mars based on two criteria: 1) minimize the time of flight to Mars; and 2) maximize the payload delivered to Mars. The designer is presented trajectories with the following hypothetical characteristics:

Table 4.1 Hypothetical Earth-Mars trajectory characteristics

Trajectory	Time of Flight (days)	Payload Delivered (kg)
1	83	242
2	209	590
3	149	310
4	191	312
5	108	363
6	175	391
7	163	500

8	256	500
9	216	367
10	119	192

Examining trajectory number 1, our mission designer would quickly conclude that the trajectory is non-dominated and therefore Pareto optimal. Trajectory 1's time of flight is only 83 days – the lowest value amongst all trajectories presented. This being the case, it is not possible for trajectory 1 to be dominated by any other trajectory in the set, thus non-dominance is quickly deduced. Similarly, trajectory number 2 is readily isolated as being non-dominated due to the fact that it has the highest payload delivery figure of all trajectories being examined. Since it cannot be inferior to any other trajectory in this criterion, it cannot be categorically inferior to any other trajectory and it too must be non-dominated.

A systematic examination of the ten trajectories reveals trajectories 5 and 7 to be non-dominated as well. Only trajectory 1 is superior to trajectory 5 in the time of flight criterion, but trajectory 5 delivers a greater payload. Trajectory 5 is not categorically dominated by any other solution, and therefore is non-dominated. Trajectory 7 is superior to all but trajectory 2 in the payload delivered category, but arrives at Mars in less time – it too is non-dominated.

All other trajectories in the set are dominated by one of these solutions. Trajectories 1, 2, 5, and 7 constitute the non-dominated or Pareto optimal solutions for the entire set; however, the remaining solutions can be further categorized. The four non-dominated trajectories can be extracted from the initial population and designated as the first Pareto front of solutions, or "rank 1" solutions. These then are the trajectories with the highest level of Pareto optimality. The remaining subset of dominated solutions can then undergo the same sorting process as before to determine further levels of Pareto optimality (rank 2, 3, etc.) within. Figure 4.1 below illustrates the trajectories with their characteristics plotted in objective space after completing the sorting and ranking process. Three Pareto fronts are seen to exist within the group of trajectories. Each rank 2 solution is dominated by at least one solution in the set of rank 1 trajectories. Rank 3 solutions possess dominating counterparts in both ranks 1 and 2. Rank 1 trajectories by definition are not dominated by any other trajectory in the overall set of trajectories.

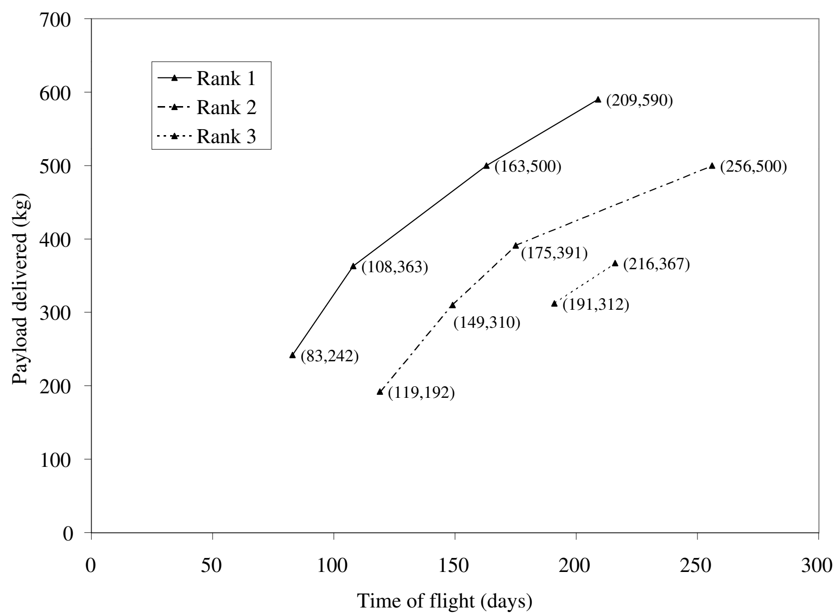

Figure 4.1 Pareto fronts for the set of hypothetical trajectories

4.3 The Pareto Genetic Algorithm

The algorithm devised in this study is based on the concept of non-dominated sorting originally conceptualized by Goldberg[23] and developed by Srinivas and Deb[12] as the Non-dominated Sorting Genetic Algorithm (NSGA). The NSGA is not an entirely new optimization algorithm, but rather a modification to the fitness evaluation procedures that exist in standard genetic algorithms. It is in some sense a supplement to a genetic algorithm that allows for a more effective means of multi-objective optimization.

The general flow of the algorithm is illustrated below Fig. 4.2. The key elements of non -dominated search, fitness assignment, and niching within Pareto fronts are highlighted and further discussed in Sections 4.3.1 through 4.3.3.

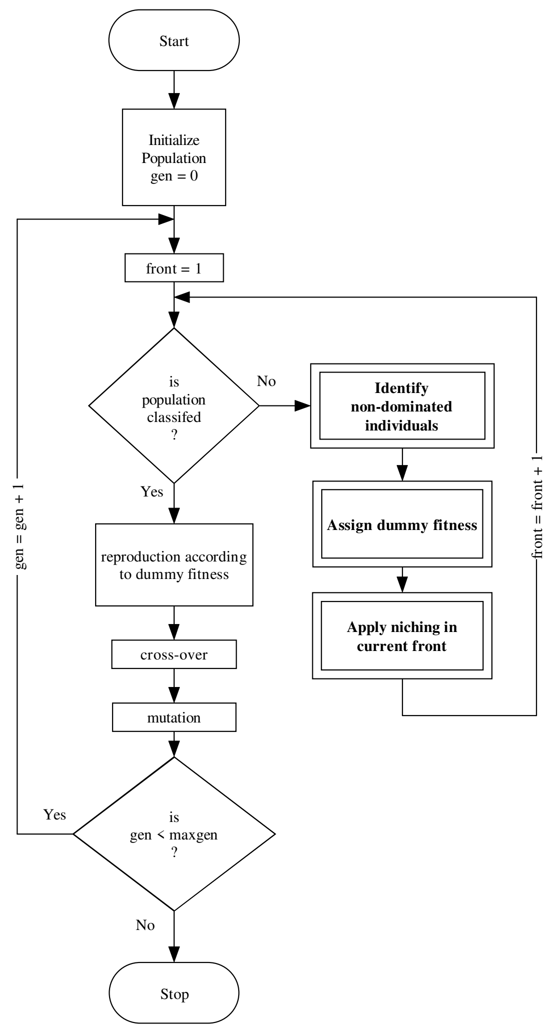

Figure 4.2 NSGA flow diagram

4.3.1 The Search for Non-dominance

The NSGA uses the previously defined concept of non-dominance to sort through a population of possible solutions, assigning each member to a Pareto optimal front according to their level of non-dominance. The process consists of three iterated steps (Fig. 4.2). The population is first sorted into dominated and non-dominated subsets. Those individuals that are non-dominated are assigned an artificial fitness according to the procedures outlined in Section 4.3.2. After assignment to a front, fitness sharing (or niching) is applied to these individuals where artificial fitness is adjusted according to solutions' proximity to one another in an attempt to evenly distribute individuals across the Pareto front. During this iteration and all subsequent ones, individuals previously tagged as non-dominated and assigned to a specific front are excluded from examination, and the remaining sub-population is examined to determine the next subgroup of non-dominated solutions. The non-dominated sorting process continues this iterative evaluation of sub-populations until all individuals in the population are assigned to a specific front and given a fitness value.

4.3.2 The Assignment of Fitness

Due to the multi-objective nature of the optimization, an artificial fitness value must be assigned to each individual in the population. An individual's fitness is associated with their level of Pareto optimality. For the first iteration of the non-dominated sorting routine, the fitness value assigned establishes non-dominated members as possessing the highest level of Pareto optimality in the population at large. This is done by initially assigning all rank 1 non-dominated solutions fitness values equal to the size of the population. A fitness value equal to the population size allows for the extreme case where each individual in the population is dominated by exactly one other member, with no two individuals being dominated by the same solution. In such an exaggerated case, each individual would constitute a Pareto front so that the number of ranks would be exactly equal to the size of the population.

After the initial fitness assignment, fitness values are adjusted within the current rank according to the niching procedures outlined in Section 4.3.3. The lowest fitness value after fitness sharing is then determined. This adjusted fitness is used as the basis for assigning the initial fitness to the next rank of solutions. It can be used as the initial artificial fitness for the next rank, or it may be decreased some small user-defined amount and the reduced value used as the starting artificial fitness for the next rank. The Pareto genetic algorithm used in this study implements the latter strategy. Use of lowest adjusted fitnesses for initial fitness assignment of subsequent ranks may promote a higher level of reproduction between ranks.[12]

4.3.3 The Importance of Niching

Fitness sharing, or niching, is as much an integral part of the Pareto genetic algorithm as the non-dominated sorting and ranking procedures. Since the goal of the multi-objective optimization is to produce a set of compromise solutions that fully represent the trades between objectives, a means must be provided for distributing solutions throughout this compromise space. This is the part that niching plays. As mentioned above, fitness sharing is applied within Pareto fronts in an attempt to promote reproduction amongst individuals in sparsely populated regions of the Pareto curve.

Niching is applied in Pareto optimization following the methods discussed in Section 2.2.4 just as it would be in any other algorithm, with the exception that it is only applied within Pareto fronts. Fitness sharing does not occur between members of different ranks. Since population members have been tagged by rank, individual fronts are easily isolated for application of fitness sharing. Within each front, the proximity of individuals to one another in the parameter space is determined using an Euclidean distance measure. Fitnesses are then degraded based upon the number of neighbors each individual has, and the proximity of those neighbors to that individual.

It should be noted that niching was conducted only in the parameter space. It would be possible to conduct niching within the objective space as well. This may in fact produce better results since an even distribution in the objective space is the desired result. Investigation of such an alternate form of niching thus is a potential area for further research.

4.3.4 The Benefits of Pareto Search

The benefits of incorporating a Pareto search algorithm in the trajectory optimization process are twofold: 1) elimination of the problems encountered in classical multi-objective optimization methodologies such as objective conflict; and 2) development of a Pareto optimal front of solutions, providing an array of compromise solutions. When applied to the population-based genetic algorithm, these Pareto concepts should enable automatic generation of Pareto optimal solutions. In the context of spacecraft trajectory optimization discussed in this study, a Pareto genetic algorithm should provide the mission designer the capability of generating "families" of Pareto optimal trajectories illustrating the trades between defined objectives.

Chapter 5

VERIFICATION OF PARETO GENETIC ALGORITHM PERFORMANCE

A rigorous testing of the Pareto genetic algorithm was first performed before proceeding to apply the technique to any low-thrust interplanetary mission optimization problems. A set of test functions comprised of those found in the literature[12] as well as some of original design were implemented in testing the NSGA. Six diagnostic test functions were used: three functions from the Srinivas and Deb test suite to test replication of performance, along with three new functions designed to test previously undemonstrated capabilities. These extended capabilities would be required for the trajectory optimization work in this study.

5.1 The Srinivas and Deb Test Suite: Test Functions *F*1, *F*2, *F*3

The first three test functions were taken directly from Ref. 12 and were used to verify algorithm performance. The algorithm developed, although based upon and expected to perform similarly to the original NSGA, is not a replica. Substitution of the stochastic remainder selection scheme[12] with stochastic universal selection, as well as the use of a sharing scheme implementing normalized parameters are two of the known variations. Other subtle variations in the construction of algorithm operators may also contribute to slight variations in algorithm performance. The testing process detailed in the following paragraphs was performed in an attempt to minimize any existing variations in performance that might be encountered. In order to facilitate comparison, values for main control parameters such as crossover and mutation rates, population size, and sharing exponent, α, were assigned the same values as those used in the original NSGA study.[12] The values of σ_{share} were modified to account for the normalized sharing scheme. The values of all control parameters used are included in Table 5.1 below.

Table 5.1 Control parameters for test functions $F1$ - $F6$

Control Parameter	$F1$	$F2$	$F3$	$F4$	$F5$	$F6$
Maximum Generation	500	500	500	500	500	500
Population Size	100	100	100	100	100	100
String Length	28	28	30	28	30	30
Probability of crossover	1.0	1.0	1.0	1.0	1.0	1.0
Probability of mutation	0.0	0.0	0.0	0.0	0.0	0.0
σ_{share}	0.005	0.005	0.001	0.002	0.035	0.018
α	2.0	2.0	2.0	2.0	2.0	2.0
Number of parameters	1	1	2	1	2	2
Number of objectives	2	2	2	2	4	4
Number of constraints	0	0	2	0	0	2

5.1.1 Test Function *F*1

The first test case [12] possesses a single independent variable with a pair of parabolas as objectives (Fig. 5.1). The goal of this test case is to determine whether the performance of the original NSGA can be duplicated in identifying the Pareto optimal region located between the minima of the two parabolas. The optimization problem is defined as,

$$\text{Minimize:} \quad \begin{aligned} f_{11} &= x^2 \\ f_{12} &= (x-2)^2 \end{aligned} \tag{5.1}$$

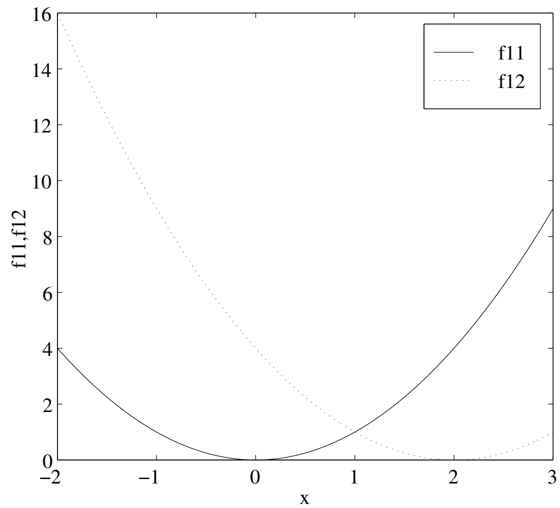

Figure 5.1 Test function $F1$

Performance was recorded statistically using the same performance measure implemented by Srinivas and Deb. This index measures the distribution of individuals along the Pareto front by dividing the front into ten sections and measuring the deviation in each section's population from the expected value of 10 (given a population size of 100). An average performance measure of approximately 2.0 for a σ_{share} value of 0.005 was expected.[12] As can be seen from Figs. 5.2 through 5.6 below, performance is similar to that obtained in the original study.

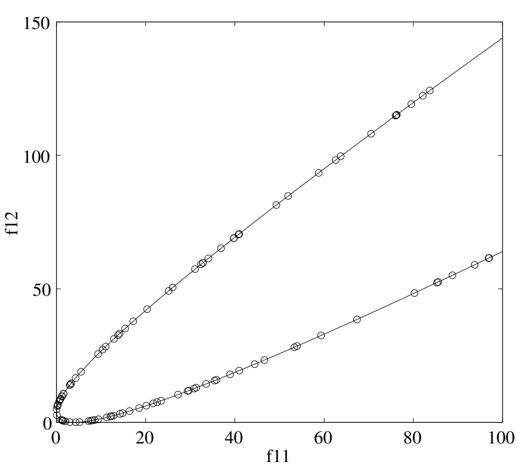

Figure 5.2 Population at generation 0 for test function $F1$

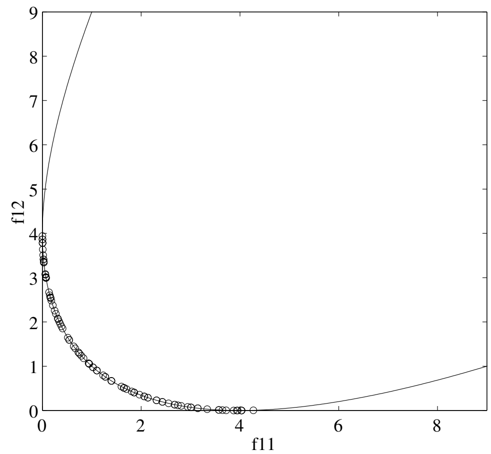

Figure 5.3 Population at generation 10 for test function $F1$

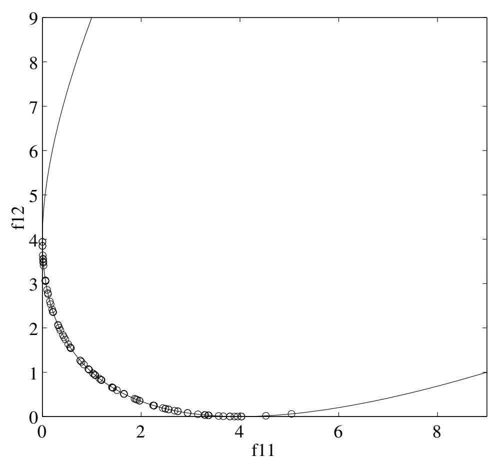

Figure 5.4 Population at generation 100 for test function $F1$

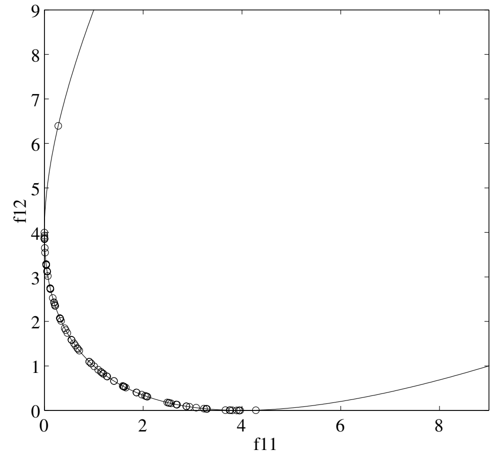

Figure 5.5 Population at generation 500 for test function $F1$

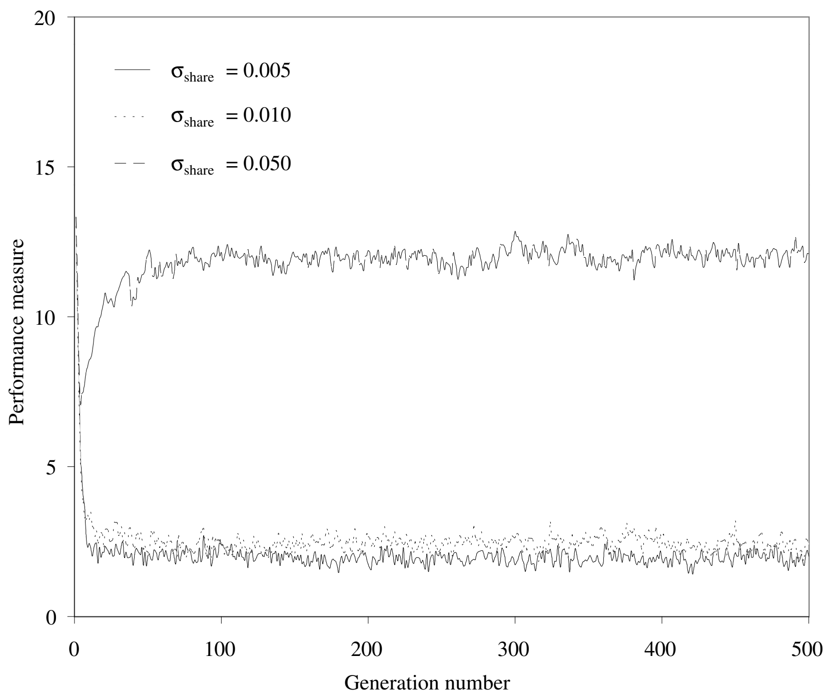

Figure 5.6 Average NSGA performance for test function $F1$

5.1.2 Test Function *F*2

The second test function [12] is comprised of one independent variable with two objectives, and is defined mathematically as,

$$
\begin{aligned}
\text{Minimize}: \quad f_{21} &= -x && \textit{if} \quad x \le 1 \\
&= -2 + x && \textit{if} \quad 1 < x \le 1 \\
&= 4 - x && \textit{if} \quad 3 < x \le 1 \\
&= -4 + x && \textit{if} \quad x > 1
\end{aligned}
\qquad (5.2)
$$

$$
\text{Minimize}: \quad f_{22} = (x-5)^2
$$

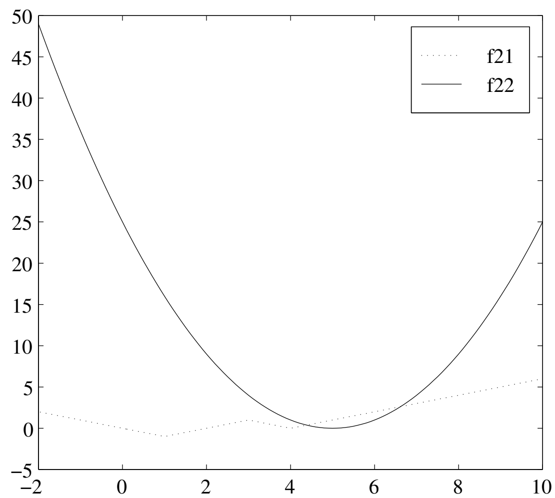

Figure 5.7 Test function $F2$

This test function (Fig. 5.7) is more complex than the first due to its disjointed non-dominated regions which reside within 1 ≤ x ≤ 2 and 4 ≤ x ≤ 5. Again, it can be seen both visibly and statistically from Figs. 5.8 through 5.12 that the algorithm is in working order. Both non-dominated regions were successfully identified, duplicating the performance seen in the original study.

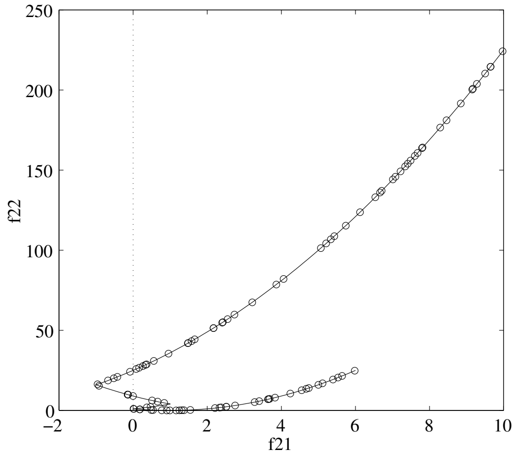

Figure 5.8 Population at generation 0 for test function $F2$

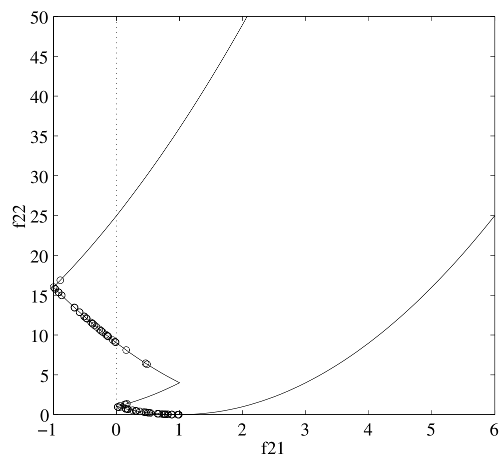

Figure 5.9 Population at generation 10 for test function $F2$

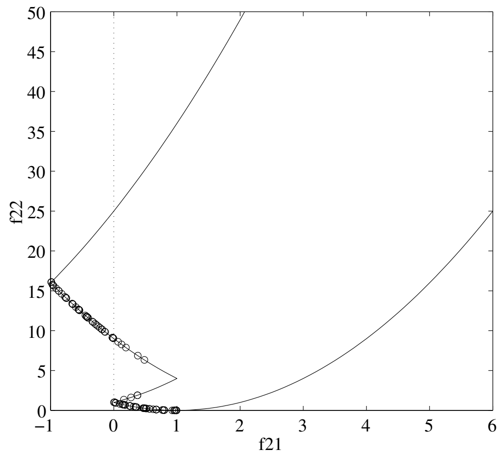

Figure 5.10 Population at generation 100 for test function $F2$

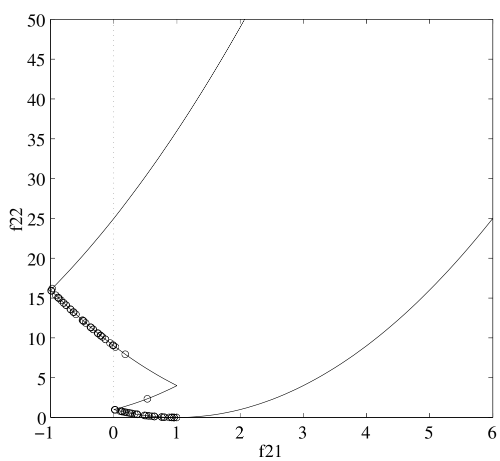

Figure 5.11 Population at generation 500 for test function $F2$

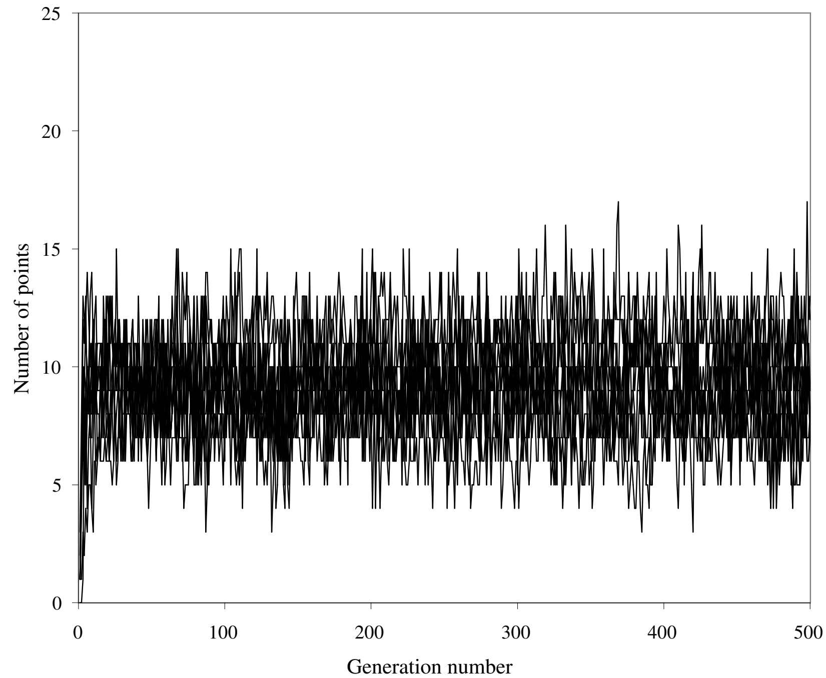

Figure 5.12 Number of individuals per sub-region versus generation for $F2$

5.1.3 Test Function $F3$

The final pre-existing test function used was the third example presented by Srinivas and Deb. This particular function was used to test the NSGA's ability to optimize multi-parameter, multi-objective problems subject to multiple constraints. The function is defined as follows:

$$
\begin{aligned}
\text{Minimize:} \quad & f_{31} = (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \\
\text{Minimize:} \quad & f_{32} = 9x_1 - (x_2 - 1)^2 \\
\\
\text{Subject to:} \quad & {x_1}^2 + {x_2}^2 - 255 \leq 0 \\
& x_1 - 3{x_2}^2 + 10 \leq 0
\end{aligned}
\qquad (5.3)
$$

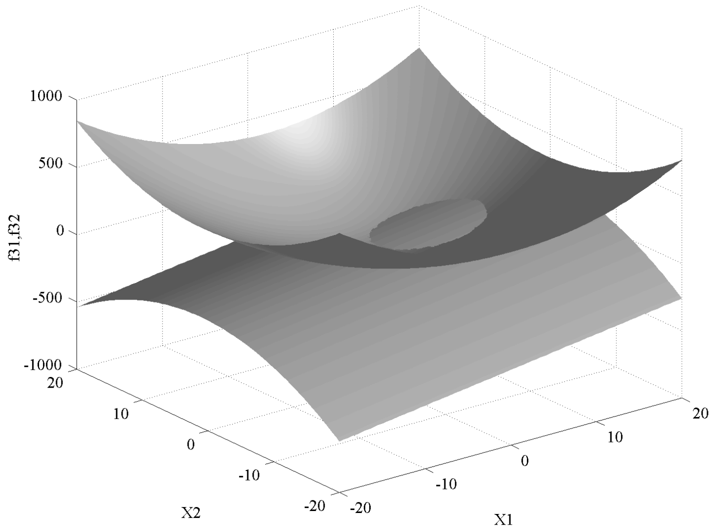

Figure 5.13 Test function $F3$

A visual representation of the unconstrained problem can be seen in Fig. 5.13 above, which reveals the objectives to be two intersecting conic sections. The problem is made more complicated by the imposition of two constraints, applied through an appended penalty function. The region of Pareto optimality can be mathematically derived as lying along the line $x_1 = -2.5$, within the region $2.5 \le x_2 \le 14.79$.[12] Replication of previous results can be seen from Figs. 5.14 through 5.16. The performance measure used in the performance analysis of test functions $F1$ and $F2$ was not applied to this test function by Srinivas and Deb, therefore no comparison performance measure statistics were recorded for this study.

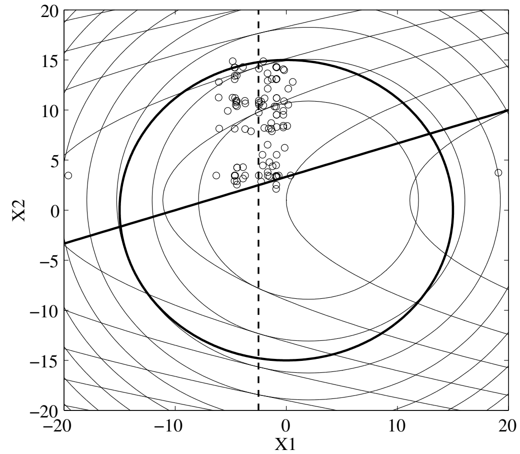

Figure 5.14 Population at generation 10 for test function $F3$

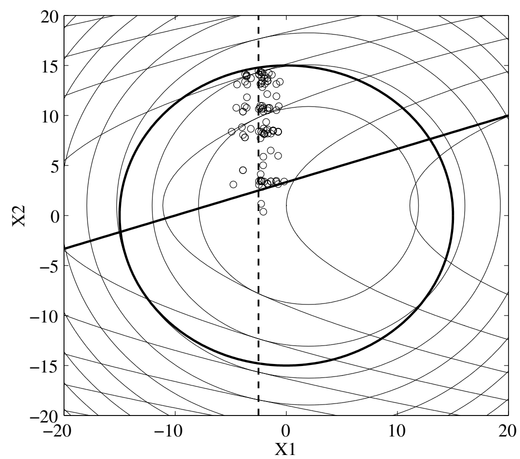

Figure 5.15 Population at generation 20 for test function $F3$

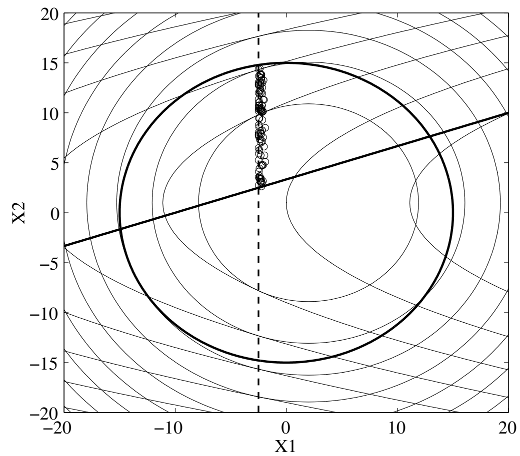

Figure 5.16 Population at generation 100 for test function $F3$

5.2 Expanding NSGA's Capabilities: Test Functions $F4$, $F5$, $F6$

The three previous test functions demonstrate the NSGA's ability to optimize with two objectives at most, each being strictly minimized. The algorithm is in fact capable of optimizing for any number of objectives, even for cases where the objective set consists of objectives of both minimizing and maximizing type. This point was noted in the inceptive article [12] as an extension for further study. Three additional test functions were devised to confirm these capabilities.

5.2.1 Test Function $F4$

The first of these additional functions maintains two slightly out of phase sinusoids as objectives. The resulting function possesses four regions of Pareto optimality, and enables the examination of the algorithm's ability to maintain more than two sub-populations of Pareto optimal individuals. The problem is defined mathematically as,

$$\text{Minimize/Maximize:} \quad f_{41} = 2\left\{\sin\left[2\pi\left(\frac{x}{5}-1\right)\right]+1\right\}$$

$$\text{Minimize/Maximize:} \quad f_{42} = \left\{\sin\left[\frac{2\pi x}{5}\right]+1\right\} \tag{5.4}$$

Furthermore, by varying which objective is maximized for optimality and which is minimized, the NSGA's a bility to handle combinations of optimization types can be investigated. Figs. 5.17 through 5.20 illustrate results for the four permutations of the problem at generation 500.

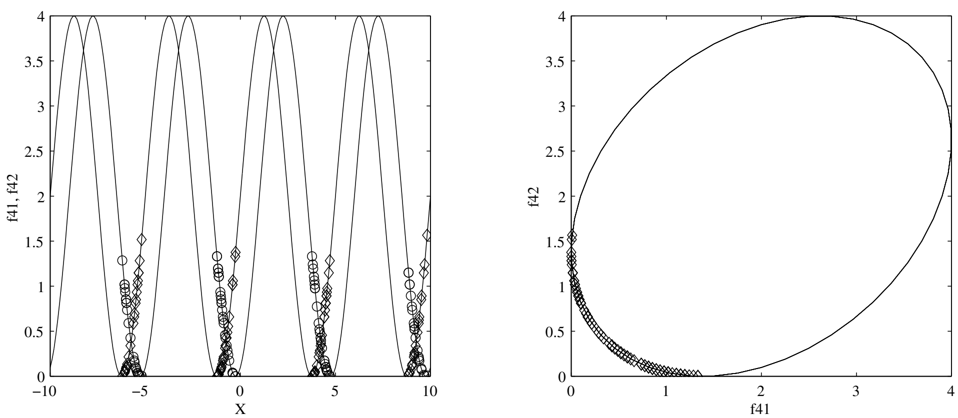

Figure 5.17 Population at generation 500 for test function $F4$ Min./Min.

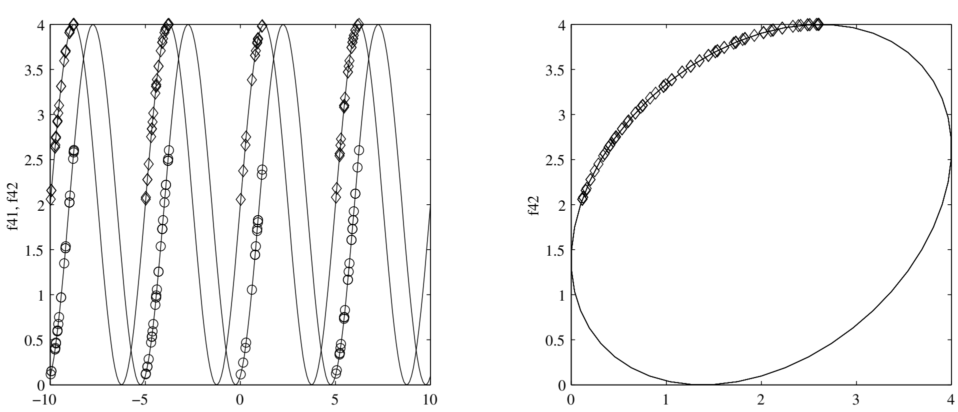

Figure 5.18 Population at generation 500 for test function $F4$ Min./Max.

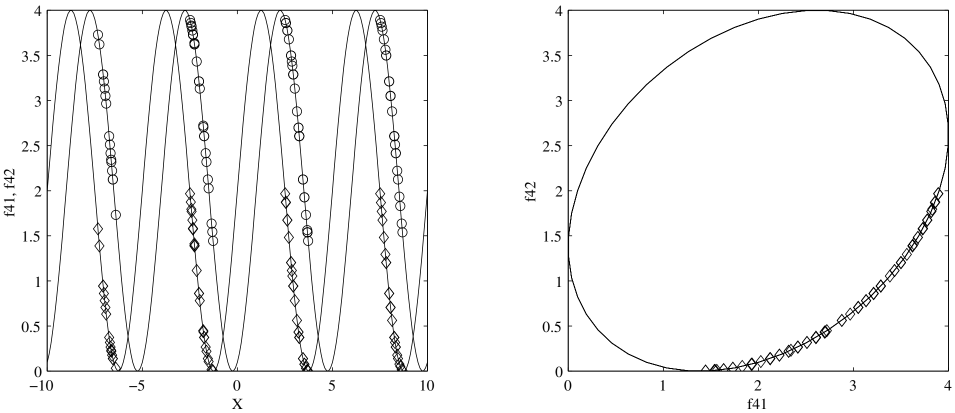

Figure 5.19 Population at generation 500 for test function $F4$ Max./Min.

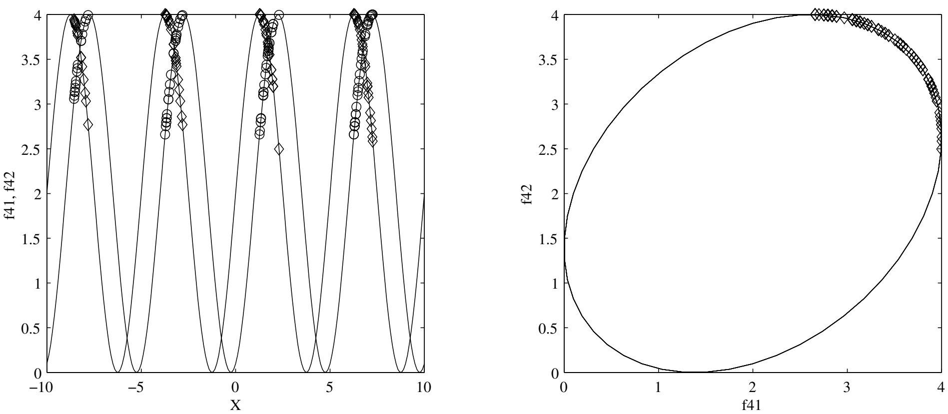

Figure 5.20 Population at generation 500 for test function $F4$ Max./Max.

5.2.2 Test Function $F5$

The second supplemental function has the purpose of verifying the NSGA's ability to handle more than two objectives. It was established that, for the purposes of the initial trajectory optimization study, no more than four objectives would be required in the multi-objective optimization. As a result, a test function consisting of four objectives was constructed to test this extended ability. Function five is mathematically described as,

$$
\text{Minimize:}\quad
\begin{aligned}
f_{51} &= (x_1-2)^2+(x_2-2)^2\\
f_{52} &= (x_1+2)^2+(x_2+2)^2\\
f_{53} &= (x_1+2)^2+(x_2-2)^2\\
f_{54} &= (x_1-2)^2+(x_2+2)^2
\end{aligned}
\tag{5.5}
$$

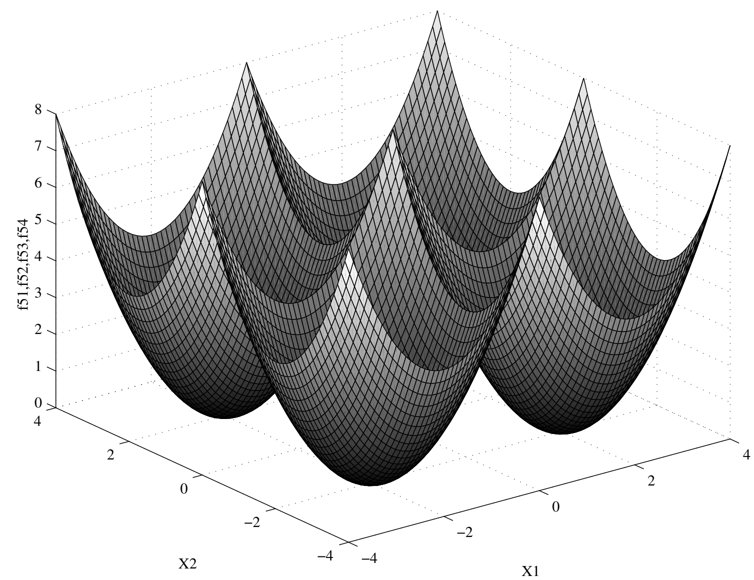

Figure 5.21 Test function $F5$

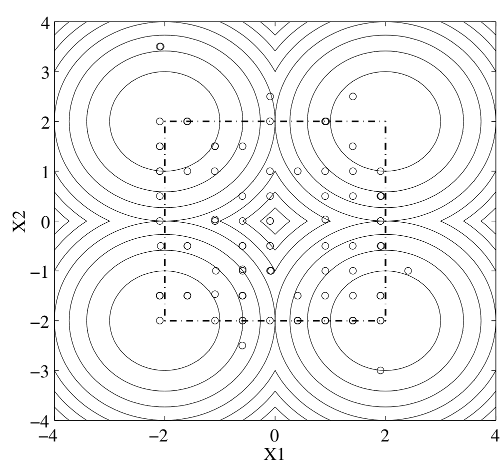

Figure 5.22 Population at generation 500 for test function $F5$

This results in four parabolic objectives (Fig. 5.21 above), whose Pareto optimal region lies within the square defined by their vertices. It can be seen from Fig. 5.22 (also above) that this region has been clearly identified by generation 500.

5.2.3 Test Function *F6*

The final function adds two constraints to the previous test function $F5$ to test higher-order multi-objective optimization with multiple constraints. The two constraints applied were similar to those applied in test function $F3$. Individuals are constrained to reside within the circle and above the line given by:

$$\begin{aligned} \text{Constraint1:} \quad & (\mathrm{x}_1 - 1.67)^2 + (x_2 - 0.5)^2 - 1 < 0 \\ \text{Constraint 2:} \quad & x_2 - 0.5x_1 > 0 \end{aligned} \tag{5.6}$$

As with test function $F3$, these constraints were applied via penalty function appended to each objective. The population at generation 500 is shown below (Fig. 5.23), illustrating the algorithm's ability to once again correctly identify the Pareto optimal region, despite the added constraints.

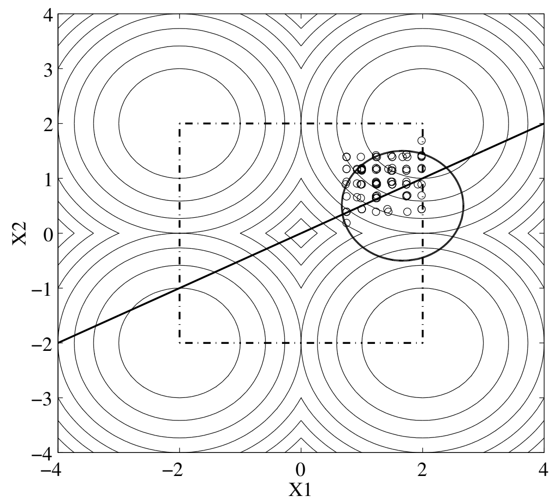

Figure 5.23 Population at generation 500 for test function $F6$

5.3 Discussion of Results

The preceding results verify that the version of non-dominated sorting algorithm developed in this studied performs accurately, reproducing the performance seen in the original study to within reasonable tolerances. Variations in performance were slight and can be accounted for by differences in coding. Additional tests verify the NSGA's extended capabilities necessary for this study. Tests to identify Pareto optimal regions for more than two objectives, as well as those to verify the NSGA's ability to handle combinations of optimization types were successful.

Chapter 6

THE NON-DOMINATED SORTING "MEMETIC" ALGORITHM

The algorithm developed for application to trajectory optimization in this study in its final form hybridizes the Non-dominated Sorting Genetic Algorithm discussed in the previous chapter with the calculus-of-variations optimizer, SEPTOP,[41] incorporating a localized search technique (Section 3.5) in the form of a Baldwinian learning strategy. As such, the algorithm falls under the category of a "memetic" algorithm – a genetic algorithm that employs a non-genetic localized search algorithm to improve individual characteristics. This designation is derived from the term "meme,"[51] a term coined by Richard Dawkins in his 1976 book, *The Selfish Gene*:

> *I think that a new kind of replicator has recently emerged on this very planet. … We need a name for the new replicator, a noun which conveys the idea of a unit of cultural transmission, or a unit of imitation. 'Mimeme' comes from a suitable Greek root, but I want a monosyllable that sounds a bit like 'gene.' I hope my classicist friends will forgive me if I abbreviate mimeme to meme. If it is any consolation, it could alternatively be thought of as being related to 'memory,' or to the French word même.*

The study of memetics theorizes that ideas can evolve in a way analogous to biological evolution. Given its memetic nature, the genetic algorithm used in this study in its final form might be termed the Non-dominated Sorting *Memetic* Algorithm, or NSMA for short.

6.1 Problem Objectives

The interface between the NSMA and SEPTOP is very much similar to that used in the previous descriptions as discussed in Chapter 3. The NSMA maintains and evolves populations of individuals representing possible trajectories, with parameters for SEPTOP being encoded as each individual's genotype. These input parameters include the co-state values associated with a given trajectory, as well as the trajectory's total time of flight. Individual fitness is evaluated through calling the SEPTOP program from within the genetic algorithm using the encoded input parameters. Each time SEPTOP is called it is run for a set number of iterations (a user-defined value), effectively executing a localized search for each member of the population. This is done in an attempt to better identify any basins of attraction that might exist in the individual's immediate vicinity.

The NSMA differs from initial hybridization attempts in its format of the computed fitness. As required by the Pareto genetic algorithm, an objective vector is maintained for each individual in the population. This objective vector contains all of the criteria of merit for the optimization, each mission objective having an associated value stored in a designated field. The fitness that is returned from SEPTOP to the NSMA is in the form of this objective vector. The fitness values stored in this objective vector are the improved fitnesses – if any improvement was seen – computed by SEPTOP after completing its run. The individual is thus assigned a new fitness value, in accordance with a Baldwinian learning strategy. The new SEPTOP input parameters are not returned as the individual's new parameter set. This is done in an attempt to maintain a greater amount of diversity in the population's gene pool, as discussed in Section 3.5.2.

The main interest of this study is to examine the trade relationship between mass delivered and time of flight for various interplanetary spacecraft trajectories. This being the case, the SEPTOP software is programmed to return payload mass and total time of flight for each trajectory in the form of an objective vector. Within the NSMA, the optimization objectives become that of maximizing mass delivered, and minimizing the trajectory's total time of flight. Thus two problem objectives exist, one of maximizing type and the other minimizing type.

6.2 Problem Constraints

The SEPTOP software attempts to derive optimal low-thrust spacecraft trajectories through application of principles from the calculus-of-variations. As was previously mentioned in Section 3.1, the measure of optimality for SEPTOP is the convergence error, a normalized measure of the extent to which all problem constraints have been met. SEPTOP attempts to drive this convergence error to zero though application of gradient descent methods. When a convergence error of zero has been obtained, an optimal trajectory has been found.

SEPTOP's convergence error is employed in the NSMA hybridization scheme as a problem constraint. The convergence error is applied to all problem objectives as an appended penalty constraint. The result of this action is to penalize each objective whenever the convergence error is nonzero, i.e. when the trajectory has not met all problem constraints.

6.3 User-Defined Controls

Restricting trajectories to meet all mission constraints while attempting to optimize their performance in terms of payload delivered and time of flight forms the requirements for mission optimization. Algorithm performance can be additionally enhanced, however, if a means is provided for the user to incorporate any prior understanding of the physical specifications of the problem. Enabling the user to bound the problem in such a manner as to narrow the search space to a region that one expects the solution to fall can greatly aid algorithm performance. Indeed it can be seen throughout this document that the more information the genetic search is provided, the more closely results match desired behavior. The user already has control over the bounds of problem parameters, namely limits on flight times and co-state values. In addition to these, the user may also wish to force the genetic algorithm to search for so-called "multi-rev" solutions, solutions that execute multiple revolutions about the sun. Forcing such solutions, while increasing time of flight, may also increase the amount of deliverable payload due to the inverse relationship between these two objectives. Forcing longer times of flights may also accomplish this goal, but without necessarily producing multi-rev solutions. This sort of gradually spiraling behavior, either in the inward or outward bound direction, has been seen through past experience to oftentimes possess optimal trajectory characteristics.

A control was thus added to allow the user to establish upper and lower bound on the allowable number of heliocentric revolutions for a given trajectory. This added control was incorporated in the form of two additional appended penalty constraints. A penalty, effectively infinite in magnitude, is applied to each objective if it so happens that a trajectory lies outside the bounded region. The magnitude of the penalty is sufficiently severe as to remove the individual solution from the upper echelons of the Pareto ranks, and to severely limit its probability of reproduction.

As was mentioned earlier, the trade relationship between the two optimization objectives, mass delivered and time of flight, is the focus of this study. Through initial trial runs of the Pareto genetic algorithm, it was found that the addition of a third objective – that of maximizing heliocentric revolutions – was necessary for obtaining viable solutions. Initial random generation by the genetic algorithm produces an initial population consisting almost entirely of individuals far from convergence (i.e. trajectories whose final states do not closely match the final boundary conditions). In initial populations, even individuals close to convergence may evaluate as dominated individuals. This is due to the fact that many trajectories exist which possess very short flight times and/or minimal thrusting. This lack of thrusting coupled with short flight time boosts objective evaluations for these solutions. Because of the weighted nature of the appended penalty function, such individuals may evaluate as Pareto optimal despite their degree of convergence. In defiance to their deficient nature, they quickly dominate the genetic algorithm population without additional control.

Use of the SEPTOP convergence error as a constraint is the supposed safeguard against such erroneous results: however, the nature of this parameter is such that decreasing amounts of information are gained from it as a solution moves further from convergence (refer to Section 3.4). As a result, using the convergence error as a constraint provides little guidance at the beginning stages of the search. In order to increase the selection pressure placed on individuals with more appropriate characteristics and avoid premature convergence, a third objective of maximizing heliocentric revolutions is added. This provides an objective inversely related to time of flight, and produces a more balanced Pareto optimization. Thus with the application of user-defined controls, the NSMA in its final form possesses three objectives (maximize mass delivered, minimize time of flight, maximize heliocentric revolutions) and three constraints (solutions must have zero SEPTOP convergence error, solutions must have a number of heliocentric revolutions greater than the lower limit, solutions must have a number of heliocentric revolutions less than the upper limit).

Chapter 7

RESULTS

Having constructed the Non-dominated Sorting Memetic Algorithm, the algorithm was subsequently applied to several low-thrust, interplanetary trajectory optimization problems. Problems ranged in difficulty level. The results from these optimizations are included in the following sections.

7.1 Control Parameters and Machine Specifics

Three test cases were run to provide proof-of-concept for the hybridized method. The control parameters used for all trajectory optimizations are listed in Table 7.1. Selection and crossover operators were implemented with no mutation. The probability of crossover was set to 1.0 with the mutation search mechanism excluded in order to observe the effectiveness of NSMA alone and maintain consistency with previous testing.[12] Niching parameters were calculated based upon the methods provided in Ref. 39. Existing methodologies for establishing σ_{share} are only rough guides however, since they require prior knowledge of the search space for each optimization problem. The value of q, the induced number of niches, was set to retain the same proportion of induced niches to population size as was used for the diagnostic test functions. For the diagnostic test functions, q was set to 10 for population sizes of 100. For the population sizes of 150 used in the trajectory optimizations, q was set to 15.

Guidance for population sizing and maximum generation determination for hybridized methods was found to be nonexistent at the time of this study, therefore population sizing and number of generations run were determined experimentally through both trial and error and limits on processing capacity. A thorough study of the effects of population size was not conducted. A population size of 150 was used as it provided a good compromise between quality of results and length of time to convergence. All simulations were run on a Sun Ultra 10 workstation with a 333 MHz ULTRA Sparc IIi processor. Run-times for the Earth-Mars flyby, Earth-Mars rendezvous, and Earth-Mercury rendezvous optimizations were approximately 16 hours, 240 hours, and 120 hours respectively, as clocked using the UNIX *time* command.

A Delta II 7925 launch vehicle was used for all simulations, along with a single 30-cm xenon engine for spacecraft propulsion. A 3.0 kW solar array was used for the flyby case, and a 5.2 kW solar array for the rendezvous case. For further details regarding engine characteristics and solar array modeling, consult Ref. 52.

Table 7.1 NSMA control parameters for trajectory optimizations

Control Parameter	Parameter Value
Population Size	150
String Length	80
Probability of crossover	1.0
Probability of mutation	0.0
σ_{share}	0.033
α	2.0
Number of parameters	8
Number of objectives	3
Number of constraints	1

7.2 Earth-Mars Flyby

The first case tested was an Earth-Mars flyby trajectory. Trajectories were constrained to 0.0 to 1.0 heliocentric revolutions. The time of flight parameter was bound between approximately two and twenty months, a time span predicted to be large enough to encompass the designated trajectory types. Launch date was fixed at September 1, 2005, a date close to an optimum ballistic launch date.

The genetic search was executed for 20 generations, resulting in the family of Pareto optimal trajectories illustrated in Figure 7.1. For the purposes of this study, the term “family” refers to a grouping of solutions in the Pareto space, each related to one another through a continuous Pareto curve. Recall that for an individual to be Pareto optimal means that no other individual has a lower cost in all three objective criteria (mass, time of flight and heliocentric revolutions). The solid line shown on Fig. 7.1 indicates the actual Pareto curve, generated by iterative calls to SEPTOP. Individuals representing converged Pareto optimal trajectories are indicated with an ×. Solutions for which trajectory plots have been included are additionally marked with the symbol , and labeled by individual number. Of the 150 members of the population, 112 were determined to be converged Pareto optimal trajectories. These 112 solutions reside in an evenly distributed manner on the actual Pareto curve, demonstrating the hybridized NSGA’s ability to discover the Pareto relationships for a given low-thrust trajectory optimization problem.

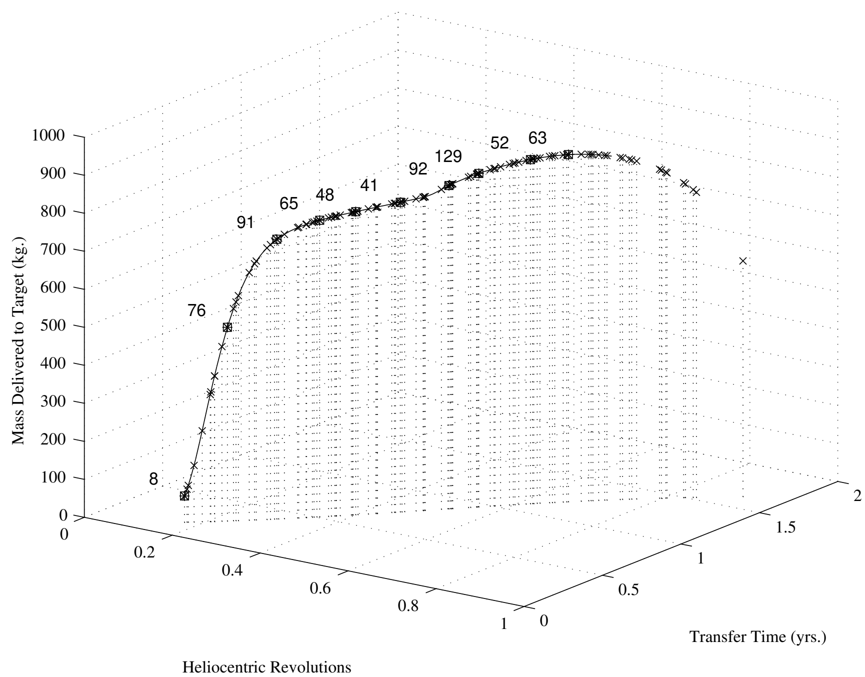

Figure 7.1 Pareto front for Earth-Mars flyby trajectory case at generation 20

The addition of the heliocentric revolutions objective, while an effective strategy for producing viable solutions, also allows for the generation of solutions which are not Pareto optimal with respect to the trade study strictly between mass delivered and time of flight. This third objective allows for the generation of trajectories with maximal revolutions, but inferior performance and times of flight. It can be seen from Fig. 7.2 that several of the trajectories generated are non-Pareto optimal with respect to the performance vs. transfer time trade. They inhabit the far right end of the time of flight spectrum (transfer time > 1.25 yrs). At this point of the graph, performance falls off as transfer time continues to increase. Trajectories in this region are of little use since alternatives exist which deliver greater payloads in less time. These solutions, however, are easily identified for exclusion from final consideration.

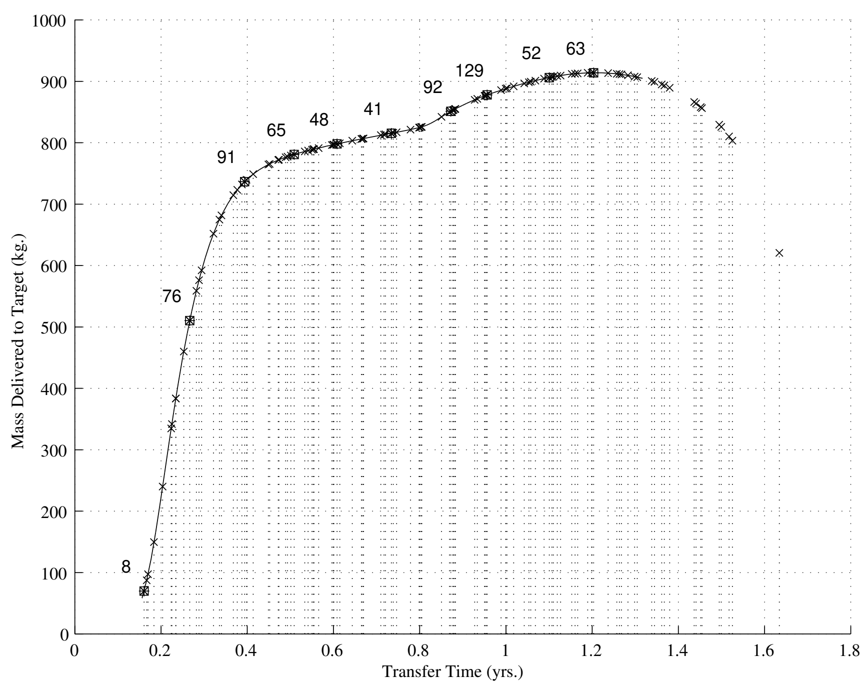

Figure 7.2 Mass delivered vs. time of flight for Earth-Mars flyby Pareto optimal trajectories

Figures 7.1 and 7.2 reveal that the optimization produced a family of solutions defining a Pareto optimal front that exists as a smooth curve in the three-dimensional objective space rather than a surface of Pareto optimal points. Figure 7.3 illustrates the indexed individuals, whose corresponding trajectories span the entire family of solutions along the Pareto curve. As is the case for all trajectory plots to follow, solid line segments represent thrusting arcs, and dashed line segments coast arcs.

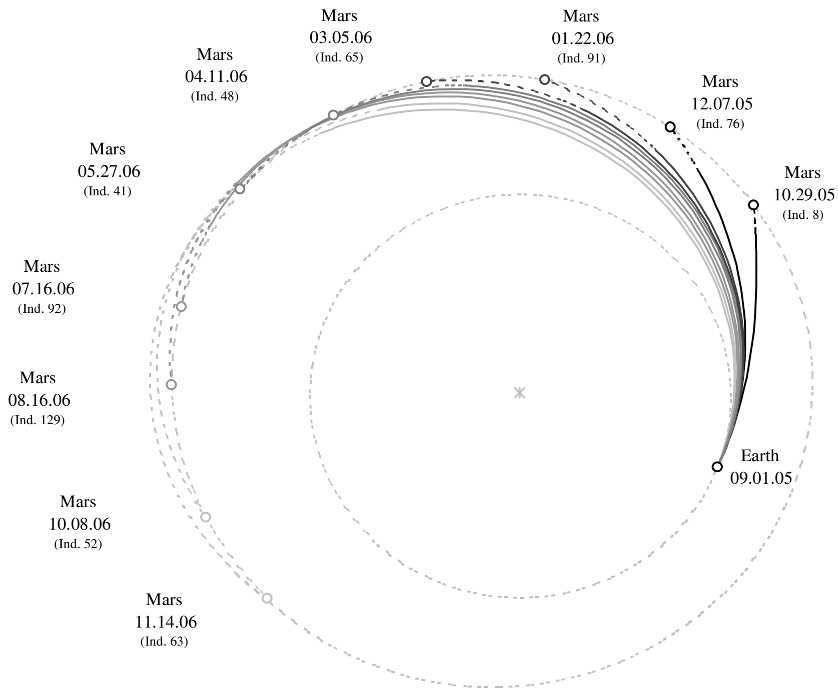

Figure 7.3 Earth-Mars flyby trajectories for indexed individuals

From these three figures, some interesting observations can be made. The Pareto front is rooted at individual 8. As transfer time increases, mass delivered rises sharply, with gains in performance tapering off as more significant thrust arcs begin to appear in the vicinity of individual 91. It can be seen from Figs. 7.1 and 7.3 that the first portion of the Pareto front - the section of the curve approximately between individuals 8 and 41 - corresponds to trajectories consisting of no more than 0.5 heliocentric revolutions, with flight paths bound by Earth and Mars orbits. Performance again picks up following the inflection point at 0.5 revolutions (0.8 yrs.), as outbound trajectories transition to inbound. The second portion of the front acquires maximum performance at individual 63 with a delivered mass of 913.96 kg. Beyond this point, performance decreases sharply, and individuals are no longer non-dominated with respect to the objectives of mass delivered and time of flight.

7.3 Earth-Mars Rendezvous

The second set of results is for the optimization of Earth-Mars rendezvous trajectories. Trajectories were restricted to 0.5 to 3.0 heliocentric revolutions, flight time bound between approximately 10 months and 3.5 years, and launch date again fixed at September 1, 2005. Figure 7.4 illustrates the population of converged Pareto optimal trajectories at generation 30. In this case, 91 of the 150 population members were non-dominated.

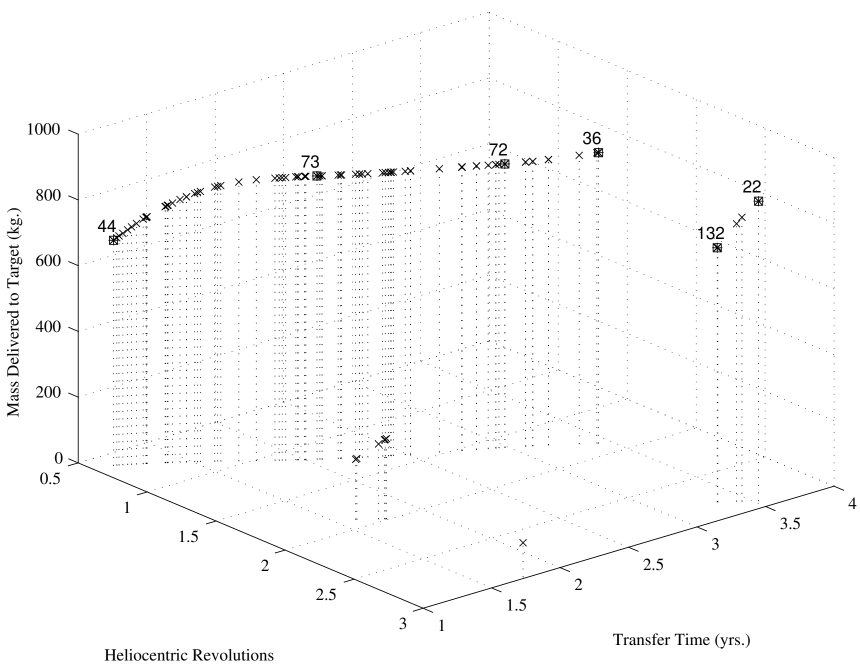

Figure 7.4 Pareto fronts for Earth-Mars rendezvous trajectory case at generation 30

Three distinct groupings of Pareto optimal individuals were identified. These were again defined by curves in the three-dimensional objective space. Figure 7.5 reveals the largest and most evenly distributed family in the Pareto space - bound by individuals 44 and 36 - to be the largely dominating subgroup. All but one of the trajectories comprising the other three families are not dominated by members of this group with respect to the m ass delivered vs. time of flight trade. As in the first trajectory case, these individuals are a small sub -population which can be easily identified for exclusion from final consideration since alternative solutions exist which deliver greater payloads in less time.

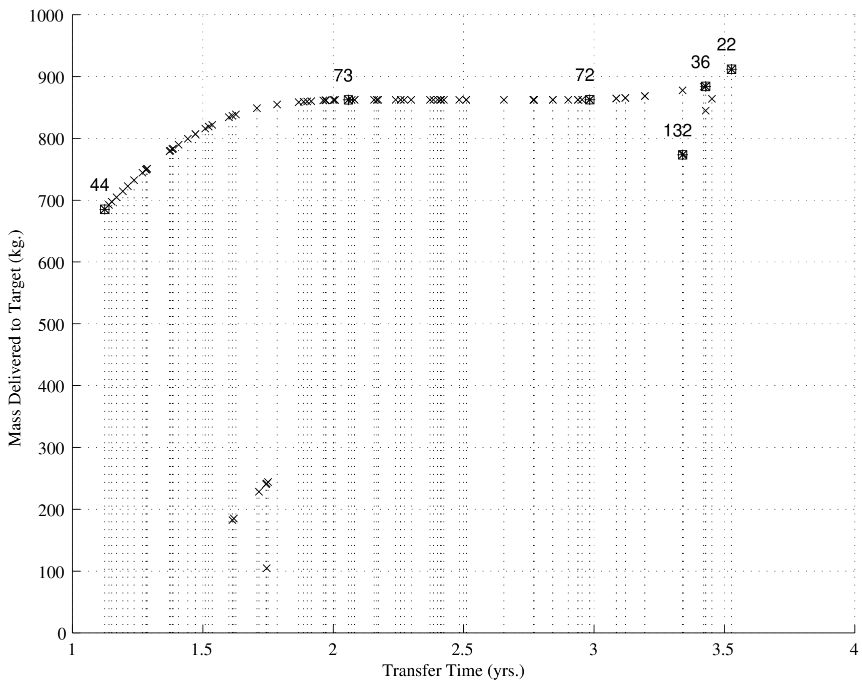

Figure 7.5 Mass delivered vs. time of flight for Earth -Mars rendezvous Pareto optimal trajectories

Beginning with individual 44 (Fig. 7.6) from the largest family with a delivery mass of 685. 28 kg, performance increases rapidly with increasing transfer time. This performance plateaus as a coast arc appears in the trajectories neighboring individual 73, and delivery masses reach approximately 862 kg. Only very small improvements in mass deliv ered are seen on the section of the curve between individuals 73 and 72. A second coast arc appears producing a burn -coast -burn -coast sequence for these solutions. In the vicinity of individual 72, an additional burn is appended creating burn -coast -burn -coast -burn trajectories and performance again rises. Performance for this subgroup of solutions approaches its maximum as trajectories transition back to a burn-coast-burn structure at individual 36 with a mass delivered of 884.10 kg.

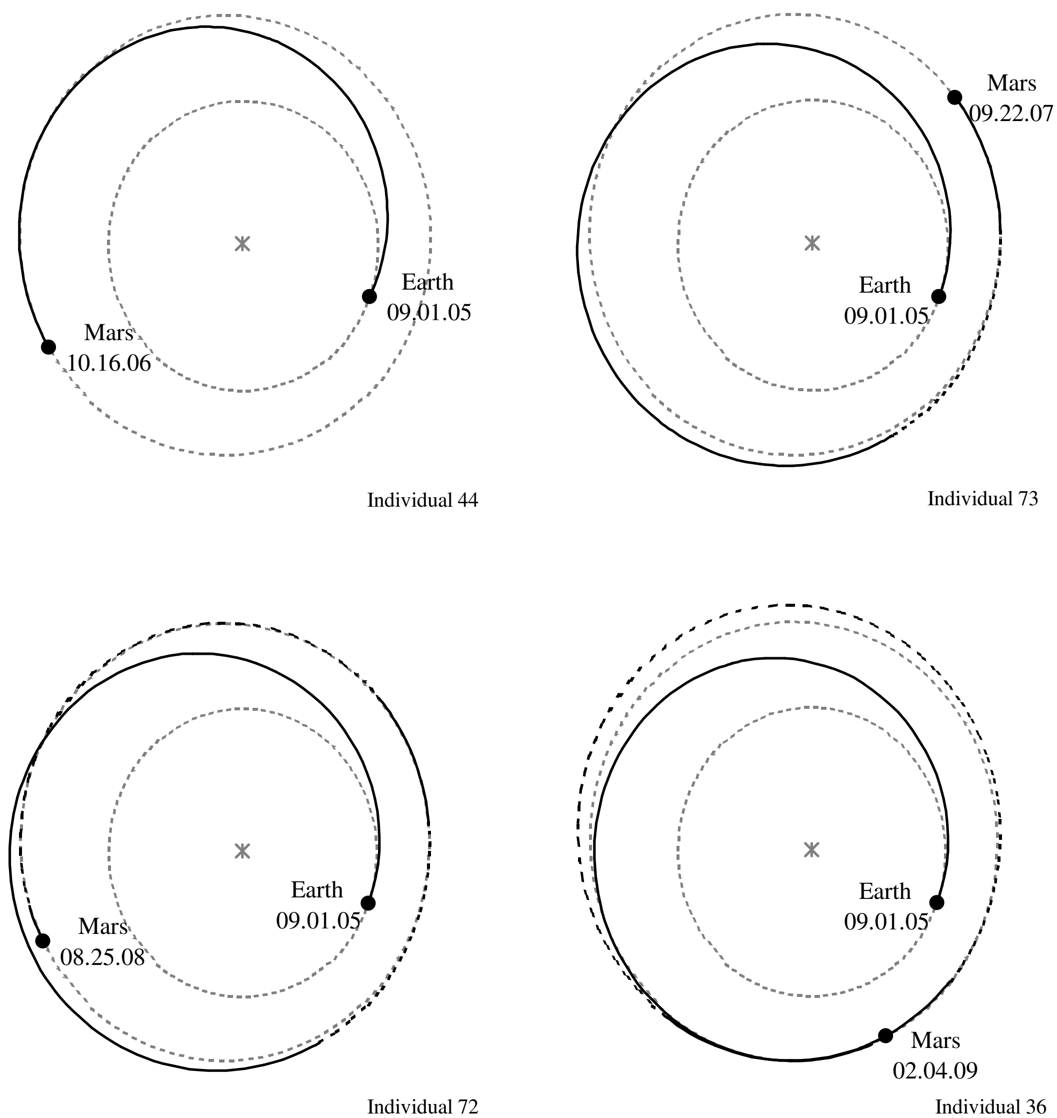

Figure 7.6 Representative trajectories for Earth-Mars rendezvous dominant family

The small family of solutions whose population resides between individuals 132 and 22 is also worthy of discussion due to the high performance and novel trajectory structure (Fig. 7.7) existent in its members. These solutions begin by taking an inward direction and spend some time performing heliocentric revolutions within or just outside of Earth orbit before spiraling out to Mars. In these subjects, two revolutions are made to increase the spacecraft's orbit inclination to more closely match that of Mars' before the trajectory progresses outward. A significant increase in performance is seen as flight times increase. The minimum mass delivered for this subset of the population is 773.33 kg (individual 132) with a maximum of 911.78 kg (individual 22) – the highest payload delivery in the population.

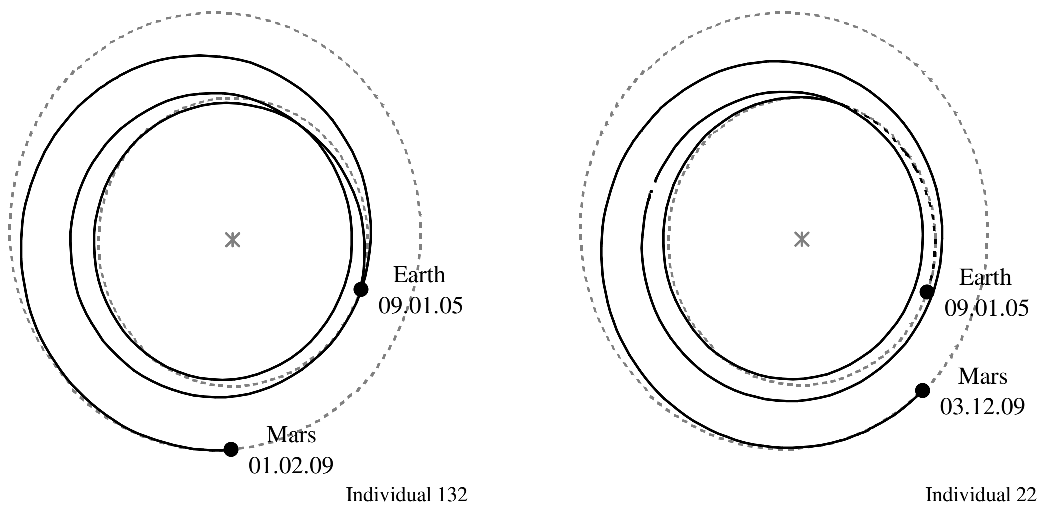

Figure 7.7 Representative trajectories for Earth-Mars rendezvous novel family

7.4 Earth-Mercury Rendezvous

The final case considered involved an Earth-Mercury rendezvous trajectory with multiple revolutions. Trajectories were specified to lie within a range of 3.5 to 6.0 heliocentric revolutions. The time of flight variable was bound to be within 550 and 915 days, and the launch date fixed at May 31, 2003. Again using a population of 150, 62 solutions were found, constituting two major families of trajectories with outlying individuals representing two other disjointed regions of Pareto optimality (Fig. 7.8). Pareto regions identified in this trial were more sparsely populated than previous cases. Figure 7.9 reveals that two of the Pareto individuals (individuals 20 and 38) are not Pareto optimal with respect to the mass delivered vs. time of flight trade study. These individuals are again easily distinguished, as they lie beneath the dominant curve visible in the population.

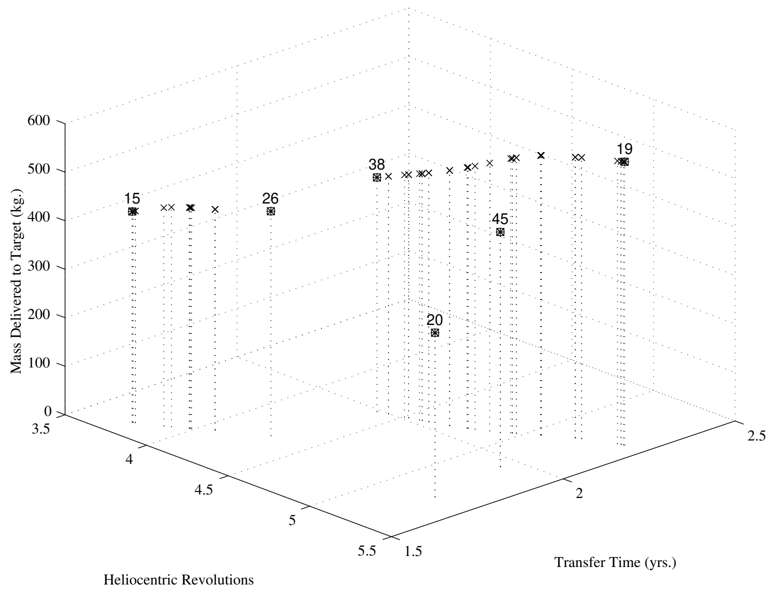

Figure 7.8 Pareto fronts for Earth-Mercury rendezvous trajectory case at generation 30

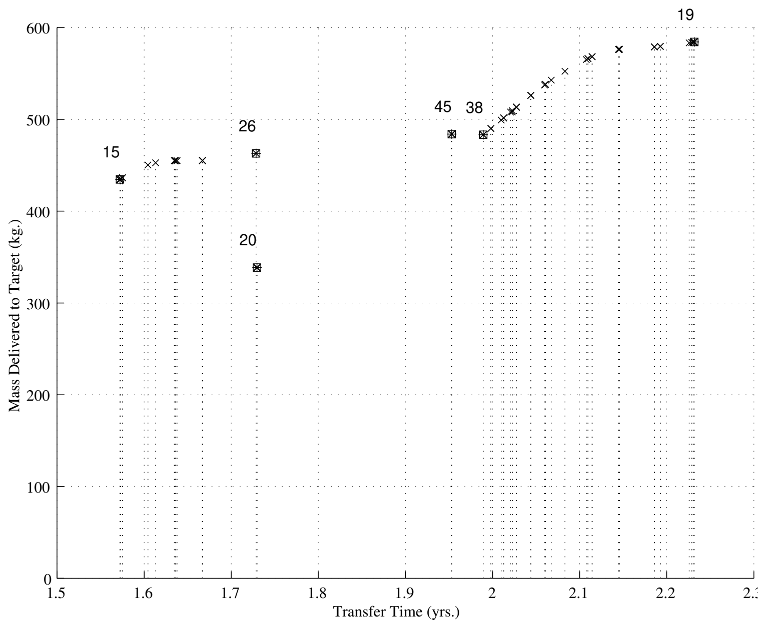

Figure 7.9 Mass delivered vs. time of flight for Earth-Mercury rendezvous Pareto optimal trajectories

All trajectories for this problem executed a gradual spiral inward to achieve Mercury rendezvous. For both of the large families in this optimization, engines were active over much of the trajectory. An initial coast arc is present for trajectories with lower flight times in both these families, with performance and time of flight increasing as coast arcs appear in revolutions closer to arrival date and thrusting moves closer to launch (Fig. 7.10). Maximum performance was achieved by individual 19, which also possessed the longest flight time and largest number of solar revolutions.

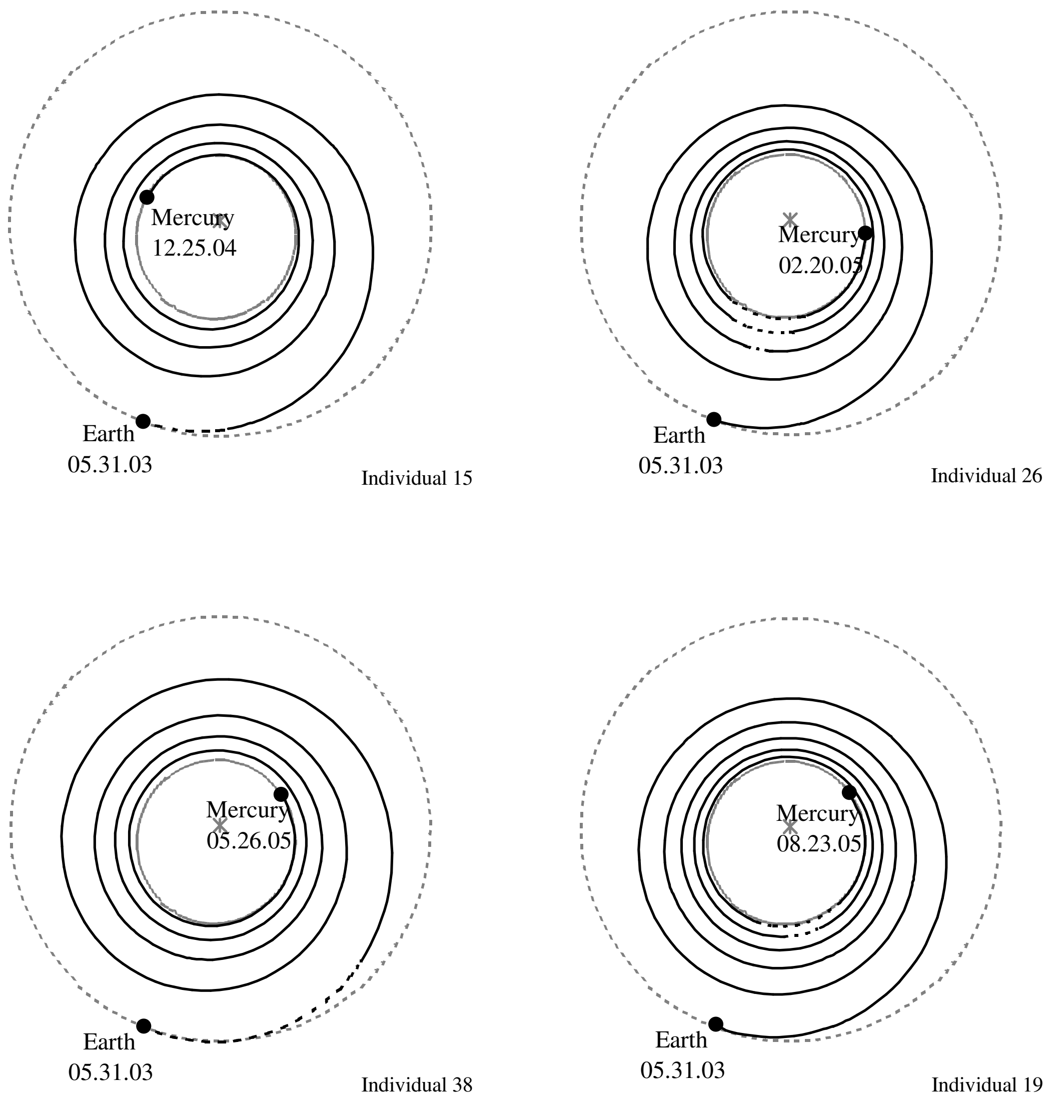

Figure 7.10 Representative trajectories for Earth-Mercury rendezvous dominant families

The isolated solutions executed more rapid descents to Mercury orbit, with larger and more frequent coast arcs than members of the larger families comparable to them in number of revolutions (Fig. 7.11). In the case of individual 20, this strategy lead to a dominated solution with respect to the mass delivered vs. time of flight trade. Individual 45, however, delivers enough mass quickly enough to qualify as a viable alternative.

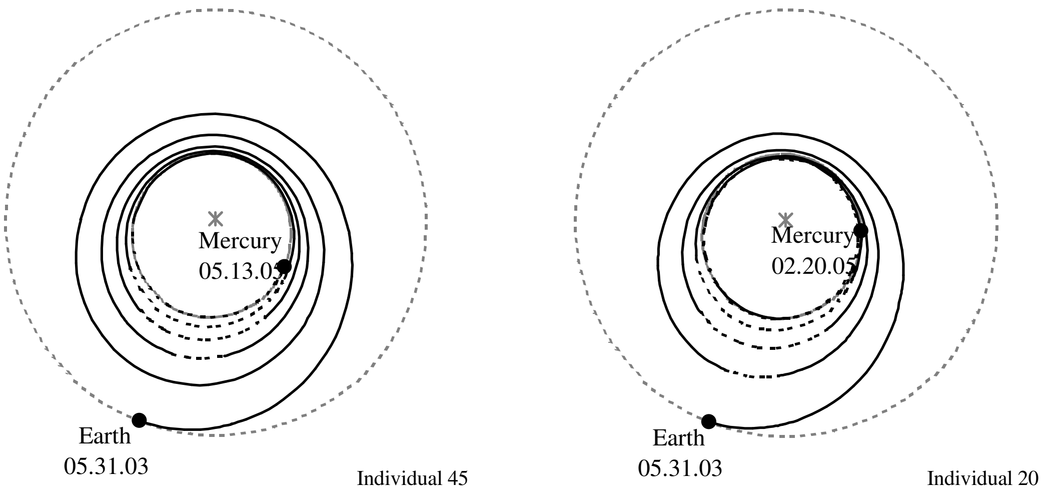

Figure 7.11 Representative trajectories for Earth-Mercury rendezvous isolated families

7.5 Earth-Mars and Earth-Mercury On-Demand

The high performance obtained by certain members of Pareto families for the Earth-Mars and Earth-Mercury rendezvous cases prompted further investigation into their potential. In the case of the Earth-Mars rendezvous, the NSMA optimization uncovered a novel trajectory class represented by individuals132 and 22. The characteristics of these novel trajectories were extracted and used to generate further interesting results. A similar process was also conducted using individuals 19 and 45 of the Earth-Mercury rendezvous population. The results are detailed in this section.

7.5.1 Earth-Mars Rendezvous On-Demand

In the case of the Earth-Mars rendezvous optimization, the control parameters for the trajectory with the highest performance (individual 22 with 911.78 kg) were extracted. The SEPTOP software was then run to compute the performance statistics for a trajectory of this type over a range of launch dates spanning approximately one Martian synodic period (778 days). For purposes of comparison, the same control parameters were also used to generate a series of trajectories over the same range of launch dates with transfer times of 1.5, 2.5, and 3.0 years.

Results of this analysis are summarized in Fig. 7.12. The solid curves in the figure represent multiple revolution SEPTOP solutions (2 and 3 revolutions) possessing flight times ranging from 2.5 to 3.55 years. The dashed curve provides a comparison with a more typical SEPTOP solution: 1.5 years and less than 1 heliocentric revolution. These curves reveal a continuous period of launch dates for a flight time of 3.55 years, all with final spacecraft mass greater than 900 kg. Shorter flight times with very large (but not continuous) launch periods are also available, such as the 2.5 year curve which has a launch period close to one year with performance greater than 900 kg.

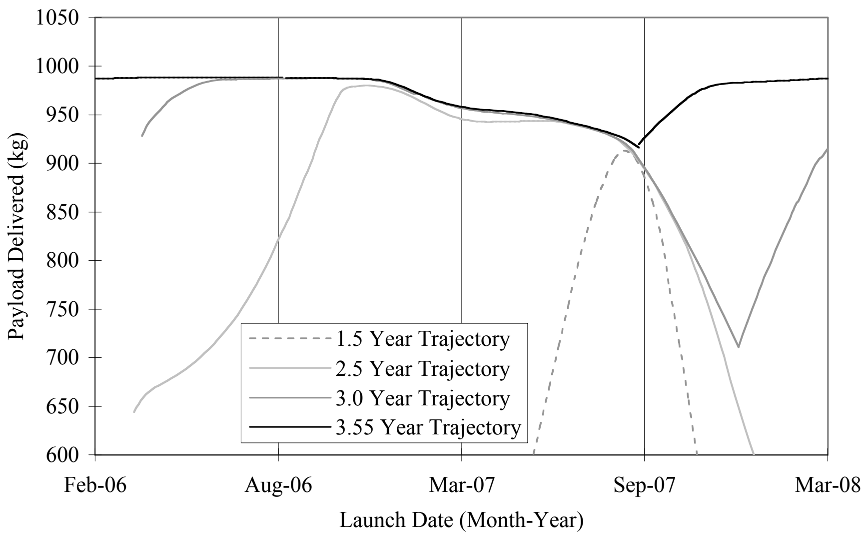

Figure 7.12 Earth-Mars rendezvous on-demand

7.5.2 Earth-Mercury Rendezvous On-Demand

Two higher performing individuals (19 and 45) from separate fronts within the Earth-Mercury rendezvous population were investigated further to determine if these trajectories could produce good performance for large ranges of launch dates as well. The period of interest is an Earth-Mercury synodic period, which is 116 days. Figure 7.13 displays the results for four different cases. Two are based on individual 19 but with total mission lengths of 2.25 and 2.0 years and two are based on individual 45 but with flight times of 1.75 and 2.0 years. The performance of the cases based on individual 45 was similar for the launch dates between mid-June to mid-July. For launch dates proceeding these few months, the 2.0-year flight time outperforms the 1.75-year trajectories. A similar trend is found for the two cases based on individual 19 where there are launch dates where the performance of both flight times only varies by a few kilograms and these dates are proceeded by months where the 2.25 year trajectories outperform the 2.0 year missions. An interesting observation is that by combining the results of individual 45 and 19 for the 2.0-year flight time, a relatively flat performance is achieved over the entire synodic period. Also, individual 19 offers higher consistent performance with the 2.25-year flight time but at the cost of longer missions. By identifying multiple solutions, the NSMA has produced missions that are readily available.

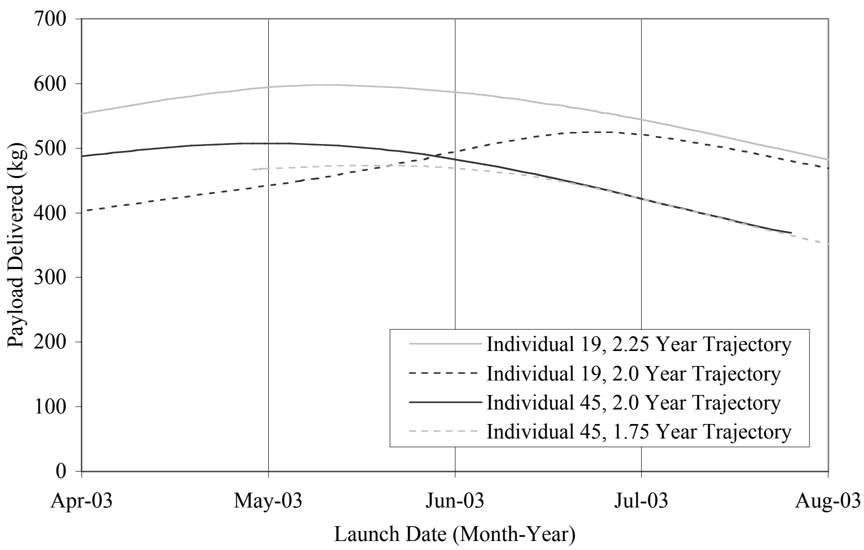

Figure 7.13 Earth-Mercury rendezvous on-demand

Chapter 8

CONCLUSIONS

In this final chapter, a summary of findings is provided along with suggestions for further research. The first section reviews the results from the application of the NSMA to the low-thrust trajectory optimization problems included in this thesis, and draws conclusions with regard to its performance. This summary is then followed by a section that suggests areas for further research, with the majority of recommendations geared towards improving algorithm speed and performance.

8.1 Summary

Based upon the results seen in this study, the hybridization of a Pareto genetic algorithm with a calculus-of-variations optimizer as a local improvement procedure proves an effective method for generating sets of optimal interplanetary low-thrust trajectories. The algorithm formulated in this thesis, the Non-dominated Sorting Memetic Algorithm, was applied to three different interplanetary trajectory optimizations: Earth-Mars flyby, Earth-Mars rendezvous, and Earth-Mercury rendezvous. Families of optimal trajectories were obtained in all test cases, with family members related through continuous Pareto curves. Best results were obtained for simple trajectories with a low number of heliocentric revolutions. As trajectory complexity increased, populations were distributed less evenly over apparent Pareto curves. These population distributions may improve with further generational cycles, discovering new portions of Pareto curves or surfaces, or filling in those partially populated at algorithm termination.

The algorithm also proved useful in producing novel trajectories. The new solutions discovered possessed both non-intuitive structures and very high performance. Unique trajectories found by the genetic search in the Earth-Mars and Earth-Mercury rendezvous cases were used to generate new and versatile trajectory classes, each possessing a continuous period of launch dates and high performance. A flat performance greater than 900 kg was achieved for a 3.55-year trajectory in the Earth-Mars rendezvous optimization. Equally important were the 3.0- and 2.5-year trajectories. These trajectory types demonstrated high performance over wide ranges of launch dates while approaching more realistic mission objectives in terms of flight-time.

The same types of continuous-launch-period trajectories were also generated for the Earth-Mercury rendezvous case using novel trajectories from the NSMA-generated solutions. Again, a relatively flat performance of approximately 500 kg was achieved for a 2.0-year trajectory. Performance approaches 600 kg as the trajectory time of flight is extended to 2.25 years. The period of launch dates for these novel trajectory classes again either approaches or extends over a full Earth-Mercury synodic period, meaning that approximately the same level of performance is available to the mission designer regardless of the date of launch.

The ability of the NSMA to generate families of Pareto-optimal trajectories was as expected. The ability of the NSMA to generate new trajectories and even new trajectory classes was hoped for, but not entirely anticipated. One of the outstanding characteristics of genetic algorithms is their ability to produce novel solutions. The absence of any preconceptions with regard to solution structure allows the GA to produce inventive solutions. Thus it was hoped that one of the benefits of applying a genetic algorithm to the foregoing trajectory optimizations would be the ability to produce new results.

The generation of trajectory classes with relatively flat performance, however, was an unforeseen and pleasantly surprising result. The generation of such trajectory classes by the NSMA provides new results in the area of low-thrust trajectory optimization. The power of stochastic optimization was thus demonstrated, as the genetic algorithm implemented in this thesis proved highly capable not only at optimizing difficult low-thrust trajectory problems, but also in generating novel, useful solutions.

8.2 Suggestions for Further Research

The main drawback of the method used in this thesis is the length of time required to execute the optimization. Algorithm run-time for even the simple Earth-Mar flyby trajectory case was on the order of 20 hours. The more difficult multi-rev rendezvous cases required on the order of 100 to 200 hours for execution. During the course of this study, several ideas for reducing execution time (beyond obvious remedies such as acquiring a faster processor or parallelizing the algorithm) were conceived and contemplated, but never implemented. The following sections detail these and other ideas for future topics of research.

8.2.1 Alternate Fitness Evaluation and Assignment

An overwhelming percentage of execution time for the NSMA is due to the localized search algorithm, SEPTOP. Each time an individual is evaluated for fitness, the SEPTOP program is executed. The time required for this execution varies, but can take up to several minutes depending upon the quality of the initially guessed set of costates and the number of iterations that SEPTOP is allowed to run. As a result, the amount of time spent executing the sorting, ranking, and genetic operations of the NSMA is minute compared to that required by SEPTOP.

One strategy for reducing run-times would thus be to target the SEPTOP software and attempt to reduce or limit its usage. The results in this thesis were obtained by setting the SEPTOP iteration limit large enough so that the algorithm would either execute until a converged trajectory had been found, or until the current solution could no longer be improved. This guaranteed adequate localized search, but resulted in decreased search efficiency when bad initial guesses were supplied. Limiting the number of iterations SEPTOP is allowed to execute would decrease run-time for localized search and requires only a simple alteration to the SEPTOP input file. Determining an appropriate number of iterations at which to terminate localized search, however, may be problematic and is most likely problem-dependent. More study is needed in this area to determine the degree of localized search that is necessary. Whether or not generalizations can be made about the extent to which localized search should be implemented is unknown. If such information is problem-dependent and cannot be generalized, perhaps some kind of automated procedure for determining localized search depth could be constructed.

An alternative to limiting localized search would be to use it only occasionally. Localized search is required to pinpoint individual fitness. If the fitness measure could instead be surmised by approximating the cost surface or through estimation using some proximity measure between existing solutions, run-times may depreciate significantly. For example, the overall algorithm would proceed at a faster rate if occasionally an individual's fitness could be approximated by copying the fitness of its closest evaluated neighbor. If SEPTOP could be bypassed and such an estimation used 10% of the time, the algorithm should theoretically run approximately 10% faster. Such methods for fitness approximation could thus also be beneficial to the method developed.

Finally, the localized search algorithm itself could be replaced. Since the NSMA can be applied using any localized search procedure, SEPTOP could be replaced by another search algorithm. A direct method could be used in place of the COV formulation. The NSMA is highly versatile in this regard. An alternate optimizer has yet to be investigated.

8.2.2 Combined Learning Strategies

As was mentioned in Section 3.4.3, several investigations into the use of partial Lamarckianism have been conducted with good results (See Refs. 42, 43, 44, 47, and 48). Implementation of partial Lamarckianism may also greatly improve algorithm speed for problems such as those presented in this thesis. Using such a strategy, an individual's fitness *and* genotype are returned after execution of localized search for some small percentage of time. This has the effect of slowly moving the search to beneficial areas of the search space. Such a gradual movement may provide faster algorithm execution since an increasing number of solutions in the genetic population will represent good initial guesses, limiting the amount of inefficient search by the localized search technique in poor areas of the search space.

Some alterations to the driver program executing the localized search method would have to be made. These alterations, however, would be straightforward and would simply return an individual's genotype along with its objective vector in a probabilistic manner. The probability of Lamarckian search would be set by the user, and could be derived by consulting Refs. 43 and 44.

8.2.3 Miscellaneous Genetic Algorithm Considerations

Several considerations regarding the genetic algorithm are also in need of further investigation. As was mentioned in Section 7.1, population sizing and control parameter optimization for this method have yet to be conducted. Studies on population sizing are needed. Research has also not yet been conducted with regard to the introduction of the mutation operator or the effects of varying the probability of crossover. Control parameter optimization for hybridized Pareto genetic algorithms is wide open as a field of research.

Investigation into the use of niching in the problem objective space rather than the problem parameter space is another area for potential research. Niching is characteristically implemented using proximity measures based on individuals' parameter values. This ensures that solutions are dispersed throughout the parameter space so as to maintain diversity and produce an effective search. In the multi-objective optimizations conducted in this thesis, niching is largely introduced in an attempt to distribute individuals evenly along a Pareto curve. Thus, although the desire is to provide an even distribution in the *objective* space, niching is still conducted in the problem *parameter* space.

In this study, niching was applied using only the time of flight parameter. Since this parameter was also one of the problem objectives (minimize time of flight), niching can be viewed as having been carried out in either the parameter space or the objective space. The problem costates were not used to compute the fitness-sharing neighborhood. Algorithm performance deteriorated when they were included. Further investigation into the question of whether or not niching should be carried out in parameter or objective space may provide helpful conclusions.

8.2.4 Advanced Missions

Finally, there exists a wealth of trajectory optimization problems that the algorithm might be applied to. This thesis provided a proof-of-concept for the NSMA methodology with respect to trajectory optimization. As a result, missions ranging in difficulty level but deemed manageable were selected for testing. Although it possesses the ability to handle such problems, the algorithm has yet to be put through the rigors of more complicated multiple-flyby trajectories.

LIST OF REFERENCES

[1] Pamadi, B., "Simple Guidance Method for Single Stage to Low Earth Orbit," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6, 1995, pp. 1420-1426.

[2] Pinon, E. III and Fowler, W., "Lunar Launch Trajectory Optimization Using a Genetic Algorithm," Paper No. 95-142, AAS/AIAA Space Flight Mechanics Meeting, February 1995.

[3] Noton, M., "Orbital Strategies Around a Comet by Means of a Genetic Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 5, 1995, pp. 1217-1220.

[4] Gage, P., Braun, R., and Kroo, I., "Interplanetary Trajectory Optimization Using a Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 43. No. 1, 1995, pp. 59-75.

[5] Rauwolf, G. and Coverstone-Carroll, V., "Near-optimal Low-thrust Orbit Transfers Generated by a Genetic Algorithm," *Journal of Spacecraft and Rockets*, Vol. 33, No. 6, 1996, pp. 859-862.

[6] Coverstone-Carroll, V., "Near-Optimal Low-Thrust Trajectories via Micro-Genetic Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 196-198.

[7] Jo, J. and Prussing, J., "Necessary and Sufficient Conditions for Optimal Control Problems with Scalar Terminal Constraint," Paper No. 98-163, AAS/AIAA Space Flight Mechanics Meeting, Monterey, CA., February, 1998.

[8] Jo, J. and Prussing, J., "Necessary and Sufficient Conditions for Optimal Control Problems with Multiple Terminal Constraints," Paper No. 98-164, AAS/AIAA Space Flight Mechanics Meeting, Monterey, CA., February, 1998.

[9] Fukunaga, A., Chien, S., Mutz, D., Sherwood, R., and Stechert, A., "Automating the Process of Optimization in Spacecraft Design", *1997 IEEE Aerospace Conference Proceedings*, Vol. 4, 1997, pp. 411-427.

[10] Hakim, S., Fuchs, M., "Optimal Acutator Placement with Minimum Worst Case Distortion Criterion", *AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, & Materials Conference – Collection of Technical Papers*, Vol. 3, AIAA, New York, NY, pp. 2010-2017.

[11] Haerting, A., Meixner, H., "Combinatorial Optimization of Long-term Maneuver Sequences Applied to Geostationary Orbit Control", *Advances in the Astronautical Sciences*, Univelt Inc., San Diego, CA, Vol. 84, 1993, pp. 515-528.

[12] Srinivas, N. and Deb, K., "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, Vol. 2, No. 3, 1995, pp. 221-248.

[13] Hans, A. E., "Multicriteria Optimization for Highly Accurate Systems," *Multicriteria Optimization in Engineering and Sciences*, edited by W. Stadler, Vol. 19, Mathematical Concepts and Methods in Science and Engineering, Plenum Press, New York, 1988, pp. 309-352.

[14] Radcliffe, N, "Natural Solutions Give Their Best," *New Scientist*, Vol. 126, April 14, 1990, pp. 47-50.

[15] Betts, J., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193-207.

[16] Bohachevsky, I., Johnson, M., and Stein, M., "Generalized Simulated Annealing for Function Optimization," *Technometrics*, Vol. 28, August 1986, pp. 209-217.

[17] Matthews, R., "Hard Maths? No Problem," *New Scientist*, Vol. 148, October 28, 1995, pp. 40-43.
[18] Kirkpatrick, S., Gelatt, C., and Vecchi, M., "Optimization by Simulated Annealing," *Science*, Vol. 220, 1983, pp. 671-680.

[19] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, Vol. 21, 1953, pp. 1087-1092.

[20] Geman, S., and Geman, D., "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Proc. Pattern Analysis and Machine Intelligence-PAMI-6*, IEEE Press, 1984, pp. 721-741.

[21] Szu, H., Hartley, R., "Nonconvex Optimization by Fast Simulated Annealing," *Proceedings of the IEEE*, Vol. 75, No. 11, 1987, pp. 1538-1540.

[22] Ingber, L., "Very Fast Simulated Re-annealing," *Mathematical & Computer Modeling*, Vol. 12, No. 8, 1989, pp. 967-973.

[23] Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[24] Holland, J., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

[25] Darwin, C., *On the Origin of the Species by Means of Natural Selection, or the Preservation of Favoured Races*, London, 1859.

[26] Spencer, Herbert. *Principles of Biology*. New York, 1866.

[27] De Jong, K., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Dissertation, University of Michigan, Dissertation Abstracts International, 36(10), 5140B, 1975.

[28] Oei, C., Goldberg, D., and Chang, S., "Tournament Selection, Niching, and the Preservation of Diversity," IlliGAL Report No. 91011, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, 1991.

[29] Goldberg, D., and Deb, K., "A Comparative Analysis of Selection Schemes," *Foundations of Genetic Algorithms*, edited by G. Rawlins, Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[30] Goldberg, D., Deb, K., and Thierens, D., "Toward A Better Understanding of Mixing in Genetic Algorithms," *Journal of the Society of Control Engineers*, Vol. 32, No. 1, January 1993, pp. 10-16.

[31] Mahfoud, S., "Crowding and Preselection Revisited," *Parallel Problem Solving from Nature-PPSN2*, edited by R. Manner and B. Manderick, Elsevier Science Publishers B. V., 1992, pp. 27-36.

[32] Deb, K., and Agrawal, R., "Simulated Binary Crossover for Continuous Search Space," *Complex Systems*, Vol. 9, 1995, pp. 115-148.
[33] Deb, K., and Kumar, A., "Real-coded Genetic Algorithms with Simulated Binary Crossover: Studies on Multimodal and Multiobjective Problems," *Complex Systems*, Vol. 9, 1995, pp. 431-454.

[34] Greffenstette, J., "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 16, No. 1, 1986, pp. 122-128.

[35] Thierens, D., and Goldberg, D., "Mixing In Genetic Algorithms", *Proceedings of the Fifth International Conference on Genetic Algorithms*, edited by S. Forrest, Morgan Kaufmann Publishers, San Mateo, CA, 1993, pp. 38-45.

[36] Harik, G., Cantú-Paz, E., Goldberg, D., and Miller, B., "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations," *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*, edited by T. Back, IEEE Press, New York, 1997, pp. 7-12.

[37] Goldberg, D., "Sizing Populations for Serial and Parallel Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp. 70-79.

[38] Goldberg, D., Deb, K., and Clark, J., "Genetic Algorithms, Noise, and the Sizing of Populations," *Complex Systems*, Vol. 6, 1992, pp. 333-362.

[39] Deb, K., and Goldberg, D., "An Investigation of Niche and Species Formation in Genetic Function Optimization," *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp. 42-50.

[40] Goldberg, D., and Richardson, J., "Genetic Algorithms with Sharing for Multimodal Function Optimization," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic algorithms*, 1987, pp. 41-49.

[41] Sauer, C., "Optimization of Multiple Target Electric Propulsion Trajectories," Paper No. 73-205, *AIAA 11th Aerospace Sciences Meeting*, Washington, D.C., January 1973.

[42] Houck, C., Joines, J., and Kay, M., "Utilizing Lamarckian Evolution and the Baldwin Effect in Hybrid Genetic Algorithms," NCSU-IE Technical Report 96-01, Meta-Heuristic Research and Applications Group, Department of Industrial Engineering, North Carolina State University, 1996.

[43] Orvosh, D., and Davis, L., "Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints," TCGA Report No. 03921, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa, 1993.

[44] Orvosh, D., Davis, L., Cox, A., and Yuping, Q., "A Genetic Algorithm for Survivable Network Design," TCGA Report No. 03871, The Clearinghouse for Genetic Algorithms, University of Alabama, Tuscaloosa, 1993.

[45] Lamarck, J., *Philosophie Zoologique*, Paris, 1809.

[46] Baldwin, J., "A New Factor In Evolution," *American Naturalist*, Vol. 30, pp. 441-451, 1896.

[47] Whitley, D., Gordon, S., and Mathias, K., "Lamarckian Evolution, The Baldwin Effect, and Function Optimization," *Parallel Problem Solving from Nature-PPSN3*, edited by Y. Davidor, H. Schwefel, and R. Manner, Springer-Verlag, 1994, pp. 6-15.

[48] Houck, C., Joines, J., Kay, M., and Wilson, J., "Empirical Investigation of the Benefits of Partial Lamarckianism," *Evolutionary Computation*, Vol. 5, No. 1, 1996, pp. 31-60.
[49] Rao, S. S., *Optimization Theory and Application*, Wiley Eastern Limited, New Delhi, 1991.

[50] Dean, E., "Multiobjective Optimization from the Perspective of Competitive Advantage" [online], NASA Langley Research Center, Available from: http://mijuno.larc.nasa.gov/dfc/mdo/moo.html [cited December 3, 1998], 1998.

[51] Dawkins, R., *The Selfish Gene*, Oxford University Press, Oxford, 1976.

[52] Williams, S., and Coverstone-Carroll, V., "Benefits of Solar Electric Propulsion for the Next Generation of Planetary Exploration Missions", *The Journal of the Astronautical Sciences*, Vol. 45, No. 2, April-June, 1997, pp. 143-159.