INTERPLANETARY TRAJECTORY OPTIMIZATION WITH AUTOMATED FLY-BY SEQUENCES

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Emily Doughty

December 2020

COMMITTEE MEMBERSHIP

TITLE: Interplanetary Trajectory Optimization with Automated Fly-By Sequences

AUTHOR: Emily Doughty

DATE SUBMITTED: December 2020

COMMITTEE CHAIR: Kira Abercromby, Ph.D.
Professor of Aerospace Engineering

COMMITTEE MEMBER: Eric Mehiel, Ph.D.
Professor of Aerospace Engineering

COMMITTEE MEMBER: Dylan Retsek, Ph.D.
Professor of Mathematics

COMMITTEE MEMBER: Timothy Fitzgerald
Space Exploration Technologies Corp.

ABSTRACT

Interplanetary Trajectory Optimization with Automated Fly-By Sequences

Emily Doughty

Critical aspects of spacecraft missions, such as component organization, control algorithms, and trajectories, can be optimized using a variety of algorithms or solvers. Each solver has intrinsic strengths and weaknesses when applied to a given optimization problem. One way to mitigate limitations is to combine different solvers in an island model that allows these algorithms to share solutions. The program Spacecraft Trajectory Optimization Suite (STOpS) is an island model suite of heterogeneous and homogeneous Evolutionary Algorithms (EA) that analyze interplanetary trajectories for multiple gravity assist (MGA) missions. One limitation of STOpS and other spacecraft trajectory optimization programs (GMAT and Pygmo/Pagmo) is that they require a defined encounter body sequence to produce a constant length set of design variables. Early phase trajectory design would benefit from the ability to consider problems with an undefined encounter sequence as it would provide a set of diverse trajectories – some of which might not have been considered during mission planning. The Hybrid Optimal Control Problem (HOCP) and the concept of hidden genes are explored with the most common EA, the Genetic Algorithm (GA), to compare how the methods perform with a Variable Size Design Space (VSDS). Test problems are altered so that the input to the cost function (the object being optimized) contains a set of continuous variables whose length depends on a corresponding set of discrete variables (e.g. the number of planet encounters determines the number of transfer time variables). Initial testing with a scalable problem (Branin's function) indicates that even though the HOCP consistently converges on an optimal solution, the expensive run time (due to algorithm collaboration) would only escalate in an island

model system. The hidden gene mechanism only changes how the GA decodes variables, thus it does not impact run time and operates effectively in the island model. A Hidden Gene Genetic Algorithm (HGGA) is tested with a simplified Mariner 10 (EVM) problem to determine the best parameter settings to use in an island model with the GTOP Cassini 1 (EVVEJS) problem. For an island model with all GAs there is improved performance when the different base algorithm settings are used. Similar to previous work, the model benefits from migration of solutions and using multiple algorithms or islands. For spacecraft trajectory optimization programs that have an unconstrained fly-by sequence, the design variable limits have the largest impact on the results. When the number of potential fly-by sequences is too large it prevents the solver from converging on an optimal solution. This work demonstrates HGGA is effective in the STOpS environment as well as with GTOP problems. Thus the hidden gene mechanism can be extended to other EAs with members containing design variables that function similarly. It is shown that the tuning of the HGGA is dependent on the specific constraints of the spacecraft trajectory problem at hand, thus there is no need to further explore the general capabilities of the algorithm.Your abstract goes in here

ACKNOWLEDGMENTS

Thanks to:

- My success would not have been possible without the support and encouragement from my parents, Doug and Kathy. Also my dog Pluto, who kept me company for many hours of work. I am especially thankful for my advisor, Kira Abercromby, who has contributed to my growth and accomplishments for years before this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Statement of Problem

When designing an interplanetary spacecraft trajectory, there are various mission requirements that constrain the problem, e.g., total time of flight, planet encounter, and propulsion capabilities. These requirements may be dependent on each other, which can largely increase the complexity of optimizing a problem. Spacecraft trajectory problems are complex and require an optimization algorithm that can effectively search a design space that may contain multiple local and global minima. Optimization algorithms take input design variables that are derived from requirements and systematically alter them in the search for an optimal solution [29]. Evolutionary Algorithms (EA) are fast and effective at solving a variety of problems, but due to their heuristic nature do not guarantee the result is the best solution [35]. This may be overcome if algorithm parameters and configuration are tuned, improving the performance of solving a given problem. The biggest limitation of EAs is the ability to handle variable length sets of design parameters, which occurs when the problem includes both continuous and discrete variables. Typically, an interplanetary spacecraft trajectory is evaluated with a defined number of encounter planets and related transfer time variables. For a problem with an undefined planet encounter or fly-by sequence, the resulting number of encounter planets and related transfer times will vary. Evaluating this type of problem with EAs provides mission designers with diverse trajectories that may not have been previously considered. This also expands the capabilities of independent researchers to explore new trajectories more

efficiently. An undefined sequence can largely expand the solution space depending on the number of allowed encounters. EAs may be altered to evaluate variable sets of design parameters, but due to the increased complexity of the solutions space they may not be able to converge on a good solution. Investigating the tuning of EA parameters and configurations for evaluating undefined encounter spacecraft trajectory problems can improve the optimization performance, and thus remove the limitations on mission designers when considering these problems.

1.2 Proposed Solution

The Spacecraft Trajectory Optimization Suite (STOpS) program is a collection of EAs that work collaboratively to solve multiple gravity assist (MGA) problems [26]. MGA problems describe trajectories with a defined fly-by sequence between multiple bodies. STOpS has been continuously developed and improved as Master's thesis projects with the future goal of providing an open source trajectory optimization suite that can evaluate a variety of problems and constraints. A defined encounter sequence is commonly required by optimization solvers because they work with a constant number of design variables. However, without a predefined number of planet encounters the number of these design parameters will vary. This work compares two methods that solve MGA problems with undefined fly-by sequences for their potential performance in the collaborative scheme of STOpS, also referred to as an island model [23, 6, 34]. This will improve STOpS's capability of analyzing problems with different types of design variables for early phase mission planning.

The most common EA is the Genetic Algorithm (GA) whose functions are the basis of the island model [34]. The GA will serve as the base optimization algorithm for the two methods implemented to evaluate problems with an undefined fly-by sequence.

The first method is the Hybrid Optimal Control Problem (HOCP) which combines two optimization algorithms in a nested loop to evaluate variable types separately [23]. The outer loop will generate a sequence of encounter bodies to be evaluated by the inner loop algorithm. This method allows the GA to function as designed because the variable types are separated so the number of design parameters is constant in each loop. The second method allows for variable length members with a constant population length with a hidden gene mechanism. [6] Certain variables are "hidden" or not evaluated by the cost function determined by tags or a marker gene. While members may be evaluated with a different number of design variables, the hidden genes fill in the extra space when the constant size population is required. This solution space is larger and more diverse than problems with a defined sequence, so these solvers may benefit from the collaboration of the island model.

1.3 Literature Review

Investigation of algorithm performance for spacecraft trajectory optimization is an ongoing process as new solvers and mechanisms are developed. The process of developing base code for a common algorithms is well understood so independent researchers commonly use open source code that is provided by organizations like NASA and ESA when analyzing spacecraft trajectory problems [26, 10, 28, 8]. The solvers provided can be used by others to investigate new mission, tune parameters, or as a baseline for comparison to other solvers. In 2007 NASA released the first version of the General Mission Analysis Tool (GMAT) as a convenient GUI (and C++ source code) for anyone working to develop new mission concepts or improve current missions [28, 30, 31]. Since then, GMAT has been updated with contributions from private industry and individual collaborators and the most current release is GMAT R2020a. The Advance Concepts Team (ACT) from the ESA developed the parallel

optimization scientific libraries Pygmo (Python) and Pagmo (C++) that provides the user with many optimization algorithms to apply to different problems [28, 9, 7, 44]. The Spacecraft Trajectory Optimization Suite, developed by Tim Fitzgerald, aims to serve a similar purpose but it is only available to run in MATLAB with a license [26]. In an educational setting MATLAB is easily accessible, so STOpS has been used as the base of other educational research to help implement improvements like low thrust trajectories, three body model, and conversion to Python. An overview of the differences between these programs is found in Table 1.1.

Table 1.1: Summary of Spacecraft Trajectory Optimization Program Capabilities [26, 8]

	Orbits Model	Mission Sequence	Cost Function	Optimization Scheme
(STOpS)	2B Patched Conics, (MGA), Impulsive	Defined	Single	EAs, (LS), Island Model
Pygmo/ Pagmo	2-B Patched Conics, (MGA), (MGADSM), Impulsive, Low Thrust	Defined	Single+ Multi	37 Algs, Island Model
(GMAT)	2+3B Patched Conics, MGA, MGADSM, Impulsive, Low Thrust	Defined	Single+ Multi	(BV) Targeting

The two types interplanetary trajectory problems commonly evaluated are Multiple Gravity Assist and Multiple Gravity Assist with Deep Space Maneuvers (MGADSM) [14]. The difference between these problems is the number of design variables. The addition of DSMs requires the solver to determine the optimal point to perform a burn during a given leg of a trajectory [34]. It is apparent in Table 1.1 that the orbit model used in GMAT and PAGMO can handle more types of trajectories than STOpS. However, these programs are compared for the same type of trajectory being optimized and this work will focus on interplanetary trajectories. The current cost function in STOpS is single-objective and it only optimizes the total cost or delta-V

of a given trajectory [26]. GMAT and Pygmo/Pagmo take this one step further by simultaneously optimizing additional design variables like phasing orbits or launch velocity [31, 8, 33]. Regardless if the program can handle single or multi-objective problems, they all require a defined trajectory sequence to constrain the number of design variables being optimized [26, 33, 8, 31].

In mission design, it is common to assess a potential trajectory in different stages – where early stages generate an initial guess that is used in later stages to refine the answer with more constraints. These constraints can vary depending on the mission and come from requirements. The trajectory design of the Earth orbiting Transiting Exoplanet Survey Satellite (TESS) was completed in three stages using GMAT. The first stage used a 2-body patched conics model and a multiple shooting process solver to estimate the trajectory from an Earth orbit to lunar fly-by to science orbit. The second stage solved back from the lunar fly-by to launch vehicle separation with a 3-body patched conics model including phasing orbits. The third phase is simply a propagation of the solutions from the previous phases to check constraints are not exceeded. A mission is considered to be a boundary value problem and is solved using targeting or shooting methods that works by breaking a problem up into intervals and solving the problem in portions to determine a final solution. In the case of a spacecraft trajectory problem, this can be seen as solving the different legs of a trajectory, usually separated by planet encounters [31]. For TESS, the first and second phases used a nonlinear programming (NLP) solver as a targeter to find only feasible solutions that meet mission requirements. The first phase of this analysis is similar to the test problems from Pygmo/Pagmo and STOpS because they all start in an Earth orbit. However, for actual missions the launch vehicle requirements need to be considered and the solver effectively finds a feasible solution for phasing loops and separation constraints. STOpS and Pygmo/Pagmo both aim to optimize a solution rather than find a feasible one and could serve in the place of GMAT for the first

phase if an optimal trajectory is desired. The current capabilities of STOpS and Pygmo/Pagmo do not solve problems that consider launch requirements. All phases of analysis with GMAT required defined planet encounters.

Pygmo/Pagmo requires more programming knowledge to work with compared to GMAT's user-friendly GUI, but it can provide a more optimal final solution [45, 32, 44]. The main idea of this library is to utilize multiple algorithms at once in the form of an island model, where the algorithms share solutions while running [34]. There are 18 heuristic global optimizers, 16 local optimizers, and 3 meta-algorithms that can handle different combinations of single/multi-objective cost functions and constrained/unconstrained variables using targeting or stochastic methods [8, 33]. The support for collaboration of these algorithms is built into the Pygmo/Pagmo library where the individual algorithms run asynchronously in parallel (Chapter 3.5) [34, 33, 8]. Pygmo was release in 2015 and was originally developed with Python 2.7 [33]. Pagmo was first released in 2017 and is frequently updated, the last time being in 2020 [8]. This program would serve better for the first phase of mission design due to the current configuration of provided test functions that use 2-body patched conics.

The Pygmo/Pagmo library comes with a set of scalable test problems that are commonly used to evaluate the performance of optimization algorithms [33, 8]. ESA also provides open-source code that can be used with Pygmo/Pagmo to evaluate MGA and MGADSM problems [40]. The user can either evaluate a new mission or use one of the provided problem bounds to test the performance of an optimization scheme. In 2012 after Izzo, Ruciński, and Biscani had developed the first version of Pagmo, they investigated the benefits of algorithm collaboration and the impact of migration topology design (Chapter 3.5) [34]. They used a combination of 11 problems (scalable and spacecraft trajectory) with 6 heuristic global optimizers and ran each test case

30 times. These tests showed that in general, the migration or sharing of solutions between algorithms improves the performance [34]. They concluded that the performance of an island model system depends on the combination of base algorithms and optimization problem. The migration topology defines how the base algorithms share solutions (Chapter 3.5). It was found that the base algorithm selected has the largest impact of performance of different migration topologies [34].

In a different study, Vinko and Izzo, investigated the performance of three heuristic global optimizers and 1 local optimizer [44]. This study expanded upon the previous research by focusing on the island model performance for spacecraft trajectory optimization [34]. They first looked at the performance of each algorithm in solving MGA and MGADSM problems. The test cases used were Cassini1, GTOC1, SAGAS, Cassini2, Messenger, and Rosetta and the bounds for these cases are available through the Global Trajectory Optimization Portal (GTOP) [40]. Between the available algorithms they developed eight cases, four of which were collaborative models. For Cassini1, none of the independent solvers were able to find the know best value of 4.9 km/s because of a local minimum at 5.3 km/s. Three of the cooperative methods were able to go lower than the local minimum, with one finding the best known value. For GTOC1, none of the solvers were able to locate the best-known value but the cooperative solvers still performed the best [44]. While the results for the two MGA problems may not be favorable for the majority of the solvers, they can be used as a reference of ways for change the combination of solvers for MGA problems. Overall Vinko and Izzo concluded that cooperative methods are the more desirable solvers for MGA and MGADSM problems [44]. Izzo and Ampatzis also found that approximating the solution space for a given spacecraft trajectory problem helps heuristic algorithms converge on a better solution [44].

Jason Bryan from California Polytechnic State University developed a program based on ESA's Pagmo, exploring the capabilities of pruning, dynamic restarts, and subdomain decomposition [10]. These additions helped to prevent the solver from getting stuck and handling solutions with multiple launch windows. The program uses an island model with heuristic global optimizers and local search algorithms. MGADSM test problems from GTOP are evaluated to compare the success rate of Bryan's program and Pagmo. For the problems Messenger (reduced), Cassini 2, and Messenger (full), the two solvers only found the best known solution for Messenger (reduced). In testing, no island model or algorithm parameters were altered or investigated [10]. It was noted that increasing the number of function evaluations improves the solution, but a more efficient way to do this to alter island model and algorithm parameters [29]. The unsuccessful tests may be the result of only one island model system was used for testing because it has been found that designing a model based on preliminary testing can improve results [26]. Bryan mentions that the program would benefit from the ability to handle variable planet sequences in future work [10].

STOpS, developed in 2014, has a GUI that allows the user to customize all aspects of a given spacecraft trajectory problem and the parameters of the optimization solver [26]. This MATLAB program evaluates MGA problems in a 2-body patched conics system using combinations of 5 heuristic global optimizers and 1 local optimizer in a customized island model system (Chapter 2, 3.5) [26]. The main difference between STOpS and Pygmo/Pagmo is the implementation of parallelization, that dictates whether the algorithms run asynchronously or synchronously. Unlike Pygmo/Pagmo, STOpS runs synchronously which means that the speed of the slowest algorithm will dictate the total run time [26]. As discussed in Chapter 3.5, the synchronous implementation probably does not effect overall performance of the island model, but does increase the run time. Fitzgerald's publication provides an overview of the GA, Differential Evolution (DE), Particle Swarm Optimization (PSO), and Ant Colony

Optimization (ACO) solvers that are provided [21, 13, 29, 26]. These solvers are all heuristic global optimizers and stochastic processes that are also categorized as EAs. These algorithms aim to mimic natural optimal processes like survival of the fittest, by evaluating a population of potential solutions and evolving the population based on the "fittest" members [29]. The stochastic nature of these algorithms make them a great option for spacecraft trajectory optimization where there may be multiple global optima present [29, 34].

The development of STOpS used a similar approach to that of Izzo and others with Pygmo/Pagmo [26, 34]. The EAs were verified using test cases with Ackley and Rosebrock functions to determine how different parameters effect the performance. To test the model performance with spacecraft trajectories, a simple Mariner 10 case was used where the actual mission dates and the Earth, Venus, Mercury sequence constrained the problem [26]. The actual mission cost (4.537 km/s) was determined by evaluating the reported mission transfer times into the STOpS cost function [26]. First, the algorithms were set up in a homogeneous island model and the difference in performance for different methods of sharing solutions was evaluated (Chapter 3.5). It was found that sharing fewer solutions between algorithms helps to prevent premature convergence [26]. Fitzgerald also looked at how including a local search algorithm with a heuristic global optimizer could impact performance. Similar to Izzo and others, Fitzgerald found that pairing these types of algorithms can improve the overall performance [26, 44, 34]. STOpS was then evaluated by combining multiple heuristic global optimizers and a local optimizer in three different models. Two of these models used all of the algorithms available. Using the results of individual algorithms, a third model was develop using only two of the global optimizers and one local. When the base algorithms are chosen based on previous performance, the overall model performance improves [26]. The lowest cost achieved with a solo algorithm was 4.6135 km/s and the third island model design had the lowest cost of

4.1979 km/s [26]. Overall, the island model system improves the solution of spacecraft trajectory optimization problems [34, 26]. But determining which algorithms are used and how solutions are shared between them is critical to prevent premature convergence or limitations the ability to effectively search the design space [26].

EAs are only a subset of the solvers available in Pygom/Pagmo, but as seen in STOpS they are a large focus in independent research aiming to asses the performance of optimizing spacecraft trajectory problems [8, 33, 42, 12, 10, 26, 37, 27]. Vaile, Minisci, and Locatelli investigated the ability of different EAs and meta algorithms. This work determined that the meta-algorithm found a better set of minimum cost values. With this, they experimented with collaborative methods between meta-algorithms and EAs. These methods allowed the collaborative models to perform better than the solo algorithms [43]. The performance of different EAs combined with a space pruning method was studied by Izzo and other members of the ACT using MGA problem from the GTOP [40]. They found that when evaluating multi-objective optimization problems the pruning method helped to locate more desirable areas of the design space. This helps the EAs to search more efficiently and is similar to the work with STOpS using a local search method in combination with an EA [16]. STOpS's approach was based on Ricciardi, Maddock, and Vasile where they used a combination of local search methods to solve spacecraft trajectory problems that are more constrained. Similar to tests with heuristic algorithms, they found that the collaborative methods improved the overall performance of the model [38]. Other research that applies different optimization methods to spacecraft trajectory problems only use problems that have a defined trajectory sequence so they are not directly relevant to this work.

Another global optimizer that has been extensively investigated for the application of spacecraft trajectory optimization is NSGA-II [19, 36, 46]. This algorithm is an

altered EA that can solve multi-objective optimization problems vert effectively. Izzo and Märtens looked into how NSGA-II and EAs perform in an island model to solve an actual NASA/ESA mission proposal for a tour from Earth to Jupiter [36]. They developed 10 trajectory sequences to optimize with no DSMs. This work aimed to use a large computational platform that could handle 128 algorithms running in the island model. Even though they used a large number of islands, the populations were kept small and it did not effect the performance [36]. Similar to other work, the overall performance of the island model was better than NSGA-II on its own [36]. Researchers from Indian Institute of Technology Kanpur use NSGA-II to optimize both time of flight and launch velocity for comparison to known missions values [19]. The sequences Earth-Venus-Mars (EVM), Earth-Venus-Venus-Jupiter-Saturn (EVVEJS), and Earth-Mars (EM) were evaluated and results interpreted through correlation plots of launch velocity, launch date, and total time of flight. These tests were run with a standard population size of 100 for 300 generations in order to generate a large amount of data to visualize trends in the design space [19]. With the EVM and EM they created a plot of the results to visualize how the design space of direct and fly-by trajectories compare. With the design variables of this study, the EVM vs. EM showed that the fly-by trajectories had a lower launch velocity and a higher total time of flight than the direct transfer for this mission [19].

In the works reviewed thus far, there has not been a method presented that is able to handle a spacecraft trajectory optimization problem that has dependent continuous and discrete variables. This problem was introduced in Section 1.3 and could be used to improve the capability of the first phase of mission design by providing a variety of trajectories for a given start and end body. Two methods were discovered that handle a variable number of design factors by altering individual genes of a population member and nesting solvers with EAs (Chapter 3.3, 3.4) [6, 5, 25, 24, 23].

In Jacob Englander's dissertation (University of Illinois), space mission planning is formulated as a hybrid optimal control problem (HOCP) [23].These problems contain both real-valued (launch date, flight times, magnitudes of thrust, flyby altitudes) and categorical variables (sequence of flybys). The HOCP consists of an outer loop that finds solutions to the sequence of flybys and the inner loop that optimizes the cost of a given solution [25, 23, 24]. The benefit of the two loops is that specific algorithms that are best suited for each problem type can be selected based on performance metrics that are developed from previous research results and initial testing. For the outer loop the flyby sequence must be represented as a discrete optimization problem, this is achieved by considering each planet to be a unique integer. This allows the problem to be successfully solved by integer valued or binary GAs [23]. One difficulty with the GA is that it requires a fixed length decision vector which is related to the number of flybys, and some solutions may not have the same number of encounters. To overcome this, Englander implemented a method of null genes where the outer loop decision vector of n genes has limits of $0 < n < 15$ where a number of 1-8 represents a planet and all other values represent no encounter (Table 4.6) [23]. Once the outer loop produces a solution, the inner loop optimizes cost, such as deltaV or mass. The problem is driven by constraints of flight time or flyby feasibility, which can be determined for a specific outer loop solution. There were four different solvers investigated for the inner loop: GA, PSO, DE, and a cooperative algorithm with PSO and DE. The program was run to compare to the Galileo mission, but was not able to determine the exact flyby sequence because the Lambert's solver used could not handle resonant flybys [23].

Ceriotto and Vasile also considered the multi-objective optimization problem as a HOCP where one layer finds the optimal sequence of flybys and the second finds the optimal trajectory for a given sequence [11]. The first layer uses a "complete trajectory" method that organize all possible trajectories in a tree structure. Each

solution is checked for infeasible components that helps to rule out solutions based on constraints like flyby altitude and transfer time [11]. This method is very similar to one mentioned by Englander but was deemed invalid to use for this portion of the problem in his work[23]. Once a flyby sequence is determined, the second layer of the problem utilizes an ACO algorithm. In the formulation of this specific ACO, the work space is organized like a tree where each branch represents a leg of the problem and each leaf represents a node of a different planet or transfer option. A population of "ants" is sent out to explore the work space but unlike common ACO implementation, the population members do not leave behind a flag for single legs of the trajectory the other members consider if encountered. Instead, the population members will develop solutions and assess the feasibility and cost by referencing a trajectory model (map of solution space) [11].

A different method to handle the variable fly-by sequence problem uses "hidden genes" in a GA, referred to as a Hidden Gene Genetic Algorithm (HGGA) by Abdelkhaik [6]. This method allows the algorithm to function without an inner and outer loop where members that are bound by the maximum possible design variables [6, 5]. Different parameters and evolutionary mechanisms are investigated for Cassini1 and Cassini2 problems. This work uses MGADSM problems but also reports on results of these problems evaluated without the DSMs, but with the same problem constrains. This allowed for comparison of a given trajectory with and without DSMs, but was mainly helpful for verification of the algorithm with less design variables. The "hidden gene" method allows the core function of a EA to assess a variable size of input design variables for a given cost function. The design variables are the number of encounter planets, departure date, time of flight for each leg and other variables required for DSM calculations. This research found that changing the departure date has the largest impact on the resulting cost. This was determined by plotting the trend of individual design variables with cost. The data was gathered using

the PaGMO software's GA and DE (not in collaboration) and found that the best mechanism to advance hidden genes is a stochastic method [6]. This method found that for the Cassini1 (MGA), the minimum cost for the Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS) trajectory is 11.2259 km/s. This compares to the value of 8.383 achieved by Schlueter, Fiala, and Gerdts. Other research also achieved values within thousandths of this, but all of these values were found using only the time of flight between legs of a defined trajectory to estimate cost with a solo optimizer [40]. Introducing new trajectory sequences to a search space decreases the area a given optimizer can explore [6, 5].

1.4 Structure of Paper

This paper will first introduce the orbital mechanics model used in the calculations of spacecraft trajectory optimization problems. Along with the statistical analysis to be conducted with the SAS software JMP. Next, the optimization concepts that were implemented to solve the test problems are outlined, the main focus is the Genetic Algorithm because it is the core of the methods investigated. The understanding of how heuristic algorithms function is key to understanding the results of this work and others.

The HOCP and HGGA are initially run with two test functions: Branin's function as a scalable problem for verification and a simplified Mariner 10 mission: Earth-Venus-Mercury (EVM). These tests helped to characterize and verify these solvers by comparing the results of each for two different problem types. The performance of the HOCP and HGGA with tests determine the feasibility of implementation to the island model.

The island model testing is done with the Cassini 1 MGA test function from GTOP: Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS). The limits on the design variables from GTOP are altered so that it can evaluate a undefined fly-by sequence. It was found from initial testing that only the HGGA was feasible for island model implementation. The HGGA was tested with different design variable limits for Cassini 1 along with different island model parameters like topology and migration policy.

Chapter 2

METHODOLOGY

2.1 Orbital Mechanics Model

Planning and executing interplanetary trajectories is a complex problem involving multiple factors. But the actual trajectories are derivative of the deceptively simple Newton's Laws of Motion. They show how an object enters and maintains an orbit around a body or follow an interplanetary trajectory due to the forces acting upon it which are proportional to the acceleration of the object. The term "zero-g" is often mistakenly ascribe to objects in orbit or in interplanetary trajectories. The gravitational influence of any mass exerts to infinity, albeit falls off as $\frac{1}{r_2}$. One way to utilize the natural gravitational effects of the solar system outside of Earth's orbit is planetary fly-bys. This is where an object travels very close to a planet but does not go into orbit around it; instead it experiences a change in momentum that alters its trajectory. Objects can also create an applied force in a specific direction using a propulsion system to alter their trajectory. The cost of a propulsive maneuver on a mission is termed "delta-V". [14].

In 1961 the former Soviet Union's Venera 1 flew by Venus becoming the first spacecraft to travel to another planet [18]. Throughout the 1960s and 1970s, there were many missions attempted by both the United States and the former Soviet Union. After many failures, there came success, soon there were images and data being sent from spacecraft throughout the solar system [17, 3]. In 1971, the United States successfully put the first spacecraft in orbit another planet with Mariner 9 which traveled to Mars, seen in Figure 2.1 [15].

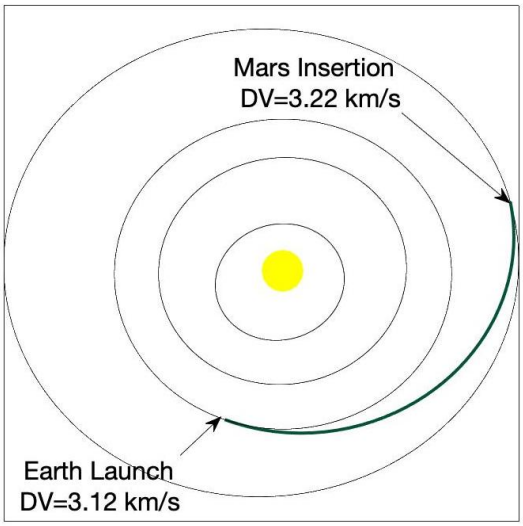

Figure 2.1: Mariner 9 Trajectory

2.1.1 Delta-V

Delta-V is the velocity required for an object to change its orbit, whether it be for a transfer or correction maneuver. For a spacecraft mission the total delta-V is a critical factor in determining the feasibility of the mission. Assuming thrust is applied with constant direction and mass; delta-V is calculated by subtracting the desired speed from the current speed of the spacecraft [14].

$$\Delta V = ||v_f - v_i|| \tag{2.1}$$

Here it is assumed the spacecraft experiences an instantaneous change in velocity that will effect the trajectory in a desired way. This is referred to as an impulsive maneuver where the spacecraft position r is not effected. There are other ways for a spacecraft to change orbit such as low thrust; however, that requires a very different way of modeling the trajectory due to the continuous applied force [14]. This work focuses on spacecraft trajectories that use impulsive maneuvers.

2.1.2 Patched Conics

The method of patched conics can be applied to estimate the total delta-V required for an interplanetary trajectory. For this system we assume a two-body orbit of the spacecraft and influencing body. In the case of interplanetary trajectories this means that when the spacecraft is outside the sphere of influence of a planet, it travels in an unperturbed orbit around the sun. The term "conics" refers to the assumption orbits are conic sections and in this case multiple conics are patched together to simulate each part of the interplanetary trajectory [14].

Looking at the example of Mariner 9 (EM) there are three conics to be considered: (1) the departure from Earth, (2) the heliocentric trajectory based on the planet positions, and (3) the arrival at Mars [15, 14]. First, the position and velocity of each planet must be found for the desired dates (planet ephemeris). Next, the heliocentric trajectory is determined based on the initial and final position of the planets using a Hohmann transfer or Lambert's solver to estimate the respective hyperbolic velocities (v_∞). A Hohmann transfer is not realistic for simulating actual missions because it requires the initial and final orbit to be circular, placing a constraint on the orbit eccentricity. The Lambert's solver relies on the assumption that solving for the velocities of a desired trajectory is independent of the orbit eccentricity given the initial and final position and the time of transfer is known. The full derivation of this solver can be found in section 5.3 of *Orbital Mechanics for Engineering Students* [14]. This solver can be described by the function

$$[v_{\infty D}, v_{\infty A}] = f(R_D, R_A, \Delta t) \tag{2.2}$$

where the subscripts D and A refer to departure and arrival and Δt is the time of transfer. Once the velocity describing each conic of the trajectory is determined, the

required delta-V of departure and arrival can be calculated

$$\Delta V_D = ||V_{P1} - v_{\infty D}||$$
$$\Delta V_A = ||v_{\infty A} - V_{P2}|| \tag{2.3}$$

$$\Delta V_{total} = |\Delta V_A| + |\Delta V_D| \tag{2.4}$$

where $P1$ and $P2$ refer to the start and end body. The total delta-V of the trajectory is the sum of these individual delta-V values [14].

2.1.3 Mission Sequence

For missions that involve more than two planets, the application of patched conics can be expanded to include the addition of planetary fly-bys or gravity assists. Spacecraft executing a planetary fly-by enters the sphere of influence of a planet but does not go into orbit or crash into the surface. Instead the spacecraft follows a hyperbolic trajectory around the planet and the resulting change in the heliocentric velocity is calculated [14]. The geometry of the flyby around a given body is shown in Figure 2.2 – the relevant variables for this model are discussed below.

To achieve the desired maneuver the main factors to assess are the difference in incoming and outgoing hyperbolic velocities and the required turn angle Beta – δ. For natural flybys the hyperbolic velocity values are equal, as seen in Equation 2.5. In this case, the turn angle is then used to determine the resulting change in the velocity vector of the spacecraft during the fly-by [14].

$$v_\infty = v_{\infty+} = v_{\infty-} \tag{2.5}$$

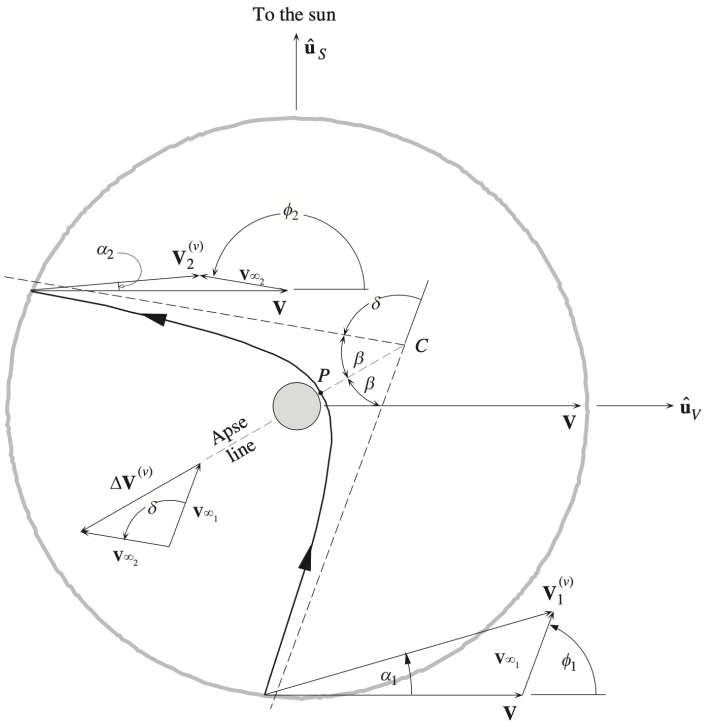

Figure 2.2: Geometry of Fly-By Trajectory [14]

$$\sin(\frac{\delta}{2}) = \frac{\mu_p}{\mu_p + r_p v_\infty^2} \tag{2.6}$$

$$\Delta V_{npf} = 2v_\infty^2 \sin(\frac{\delta}{2}) \tag{2.7}$$

Besides the velocity, the planet's standard gravitational parameter μ_p and radius of periapse during fly-by r_p are used to calculate the required turn angle. With this, Equation 2.7 describes the apparent change in velocity during the fly-by [14].

A powered fly-by is required when $v_{\infty+} \neq v_{\infty-}$ which results in either a low periapse radius or large turn angle [14]. This means that the spacecraft must apply a small impulse at the periapse radius, with the resulting delta-V calculated by finding the difference between incoming and outgoing hyperbolic velocities at the flyby periapse

radius (v_m) shown in Equation 2.9.

$$v_m = \sqrt{v_\infty^2 + \frac{2\mu_p}{r_p}} \tag{2.8}$$

$$\Delta V_{pf} = |v_{m+} - v_{m-}| \tag{2.9}$$

Combining the equations presented for a simple patched conics problem with flyby calculations produces Equation 2.10 of total delta-V required for a MGA problem [14].

$$\Delta V_{total} = \Delta V_D + \sum_{i=1}^{n} \Delta V_{pf} + \Delta V_A \tag{2.10}$$

2.2 Statistical Analysis

The SAS software JMP is used to analyze the sample sets, X, produced from the different test methods explored here [2]. For a given sample set of length n, the average or mean value, $\bar{x}$, of all the samples $x \in X$ is calculated with Equation 2.11 [20]. The mean is used to determine the spread of the samples from the standard deviation s (Equation 2.12) [20].

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) \tag{2.11}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_1 - \bar{x})^2} \tag{2.12}$$

For optimization problems, a mean that is close to the known optimal value with a low standard deviation is desired because that would indicate the solver consistently produces desired solutions. These two statistical parameters are calculated for each set as well as distributions of the data. Viewing the distributions enables a quick

qualitative assessment of differences between conditions, this characteristic is a key requirement of statistical tests for comparing means like the t-test or z-test [20]. When this criteria is not met, inferences can be made about the differences between sample sets using the distributions. Distribution are plotted as histograms (bar graphs) in which the sample set elements are assigned bins based on their value. Too few bins only yields a few bars, while too many bins leads to a spiky display – both condition obscure the shape of the distributions [20]. JMP can automatically generates the number of bins for a distribution based on the range of samples [2].

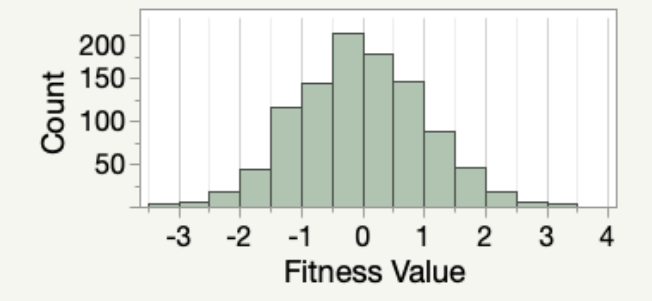

Figure 2.3: Normal Distribution

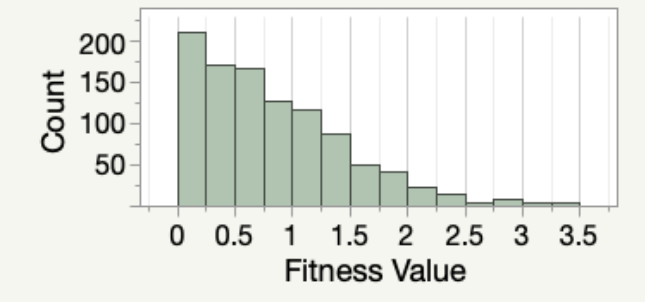

Figure 2.4: Right Skewed Distribution

Distributions tend to contain a few peaks that represent a large concentration of solutions in those bins. For a symmetric (normal) distribution a single peak will correspond to the location of the mean, but is not necessarily the case for a skewed distribution, seen in Figure 2.4. In terms of the results from an optimization problems multiple peaks may indicate the presence of local optima. Some distributions

may appear to trend to one direction (skewed) as a result of outliers present in the solutions. Considering optimization problems that aim to find a minimum, a right skewed distribution would indicate a problem with the solver because the solutions trend away from the optimum value. Where as a left skewed distribution may be acceptable because the solutions are trending towards the optimum solution.

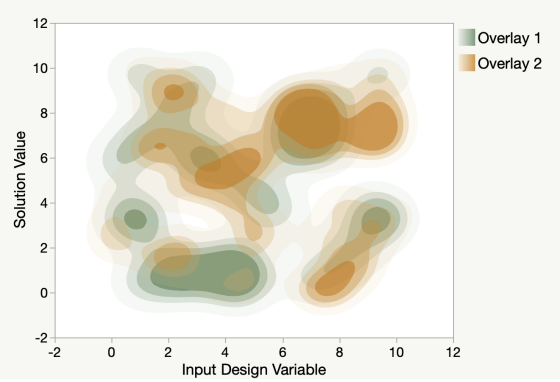

Figure 2.5: Density Plot Example

Complex solution spaces can be visualized with a contour plot where the density of solution categories is generated from pairs of design variables [20]. In JMP pairs of an input design variable and corresponding solution are plotted and the occurrence of pairs in the design space are simplified to areas with a set degree of shading related to the concentration of pairs in an area [2]. The shaded areas can represent the total set, or with the application of an overlay, additional dimensions can be visualized as shown in Figure 2.5.

Chapter 3

OPTIMIZATION

3.1 General

For any engineering problem, finding the "best" or optimal solution is always desired. When solving these problem the possibility of finding the best solution can depend on the given problem and constraints. For spacecraft missions, there are countless ways to formulate an optimization problem depending on the mission requirements considered. Some of the constraints may be derived from mission life, launch period, eclipse frequency, and propulsion capabilities [31].

Optimization is the process of varying the inputs to a given problem to find a minimum or maximum solution [29]. The solutions that are produced are referred to as the evaluated "cost" or "fitness" of a function. There may be multiple potential local optima but only one global optimum value.

An example of this concept can be see with Ackley's function that is defined as

$$f(x_i) = -a * exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos cx_i\right) + a + exp(1) \tag{3.1}$$
$$x_i \in [-32.768, 32.768] \quad for \quad i = 1, ..., d$$

where d is the number of dimensions and a, b, and c are constant values [1]. Figure 3.1 shows the 2-dimensional solution space to Ackley's function with the global optimum of $f(0, 0) = 0$. The global optimum is clearly shown in the center of the solution space at the bottom of the dark blue area and the yellow plane above this consists of

many local minima as dimples across the space. Given the potential complexities of a function's solution space, the choice of which optimization scheme to use is critical to a successful search for the global optimum [29].

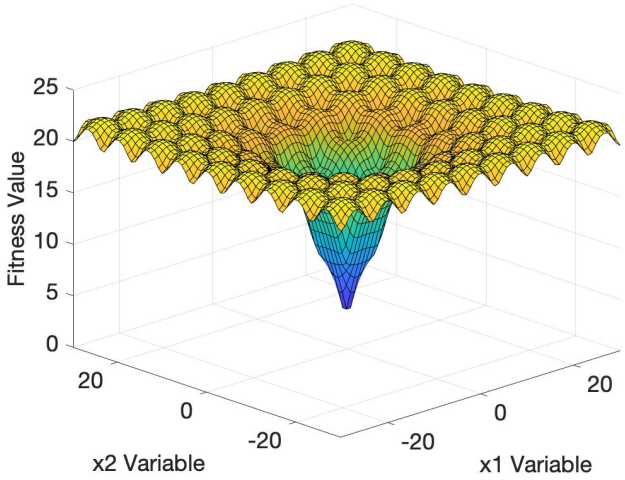

Figure 3.1: Ackley's Function 2-D

Optimization schemes have different ways of comparing the calculated cost of sets of input variables and altering those inputs to converge on the best cost [29]. The components of an optimization algorithm can be broken down into six categories. The categories outlined in Table 3.1 are introduced by Haupt in *Practical Genetic Algorithms* [29]. The factors that apply to the optimization scheme of this work will be further explained in upcoming sections.

Table 3.1: Optimization Categories [29]

1	Trial and Error	Function
2	Single Variable	Multiple Variable
3	Static	Dynamic
4	Discrete	Continuous
5	Constrained	Unconstrained
6	Random	Minimum Seeking

1. The problem to be optimized can be solved with a trial and error method or function. The trial and error method describes the process where there is a lack of knowledge of the problem in question, so inputs to the problem are varied without knowledge of whether the solution is actually improving. This method is used in experimental settings, like the development of new technology. The second option is the use of a mathematical function that describes the problem with defined variables, like Ackley's function.

2. If only one input is varied, the problem is single variable or one-dimensional. For more complex problems there are usually more than one input and this is described as multidimensional optimization. The capabilities of individual optimization schemes to handle multidimensional optimization problems differs. In the example of Ackley's function, the dimensions are defined by the user (e.g. 2-dimensions in Figure 3.1).

3. Static optimization problems output solutions that are independent of time, where as dynamic problems have solutions that are a function of time. In the Ackley example, the solution is static but in the application of spacecraft trajectory optimization time is almost always a factor.

4. Discrete variables are seen as having a finite number of possible values, like the number of cities a plane can land at. Continuous variables can have an infinite number of possible values, but are usually bounded by the constraints of the given problem. For a spacecraft trajectory problem, the number of planets to encounter is discrete and the transfer time of each leg is continuous.

5. The variables for a given problem can be generated either unconstrained or constrained. This means that the possible values the variables can take on are bounded if constrained. The optimization scheme and knowledge of the desired solution space are factors that influence applying bounds. Complex solution

spaces like spacecraft trajectory problems always use some information like launch window or planet positions to bound variables. Unconstrained problems require a specific solver are are not suited for EAs.

6. The method used to generate variables influences a solver's ability to find the global minimum of a problem. With iterative or minimum seeking processes, the variables are altered based on a determinant sequence of steps. The issue with this process is simply following steps to minimum values can cause the algorithm the converge prematurely on local minima. The other option is random processes that alters the variables with defined stochastic mechanisms, which helps introduce new areas of the solution space.

This work will focus on optimization problems that have a defined function, introduced in Chapter 2, and will be referred to as the optimization algorithm's "cost function". The complexities of a spacecraft trajectory problem with an undefined fly-by sequence requires a multidimensional solution space with constrained, dynamic variables. Stochastic processes are capable of evaluating the diverse solution space of these problems and are implemented with methods like EAs. This problem consists of both continuous and discrete variables due to the undefined fly-by sequence. The variables will be constrained by the known bounds of test functions and spacecraft trajectory test cases.

3.2 Genetic Algorithms

The most common EA, the Genetic Algorithm, aims to mimic the process of evolution found in nature [35]. This algorithm randomly generates a population composed of potential solution members. The members (chromosomes) contain a finite number of variables (genes) that are input to the cost function. The fitness, J, of each member

is evaluated and this value is used as a metric for comparison between members [29]. The relationships of these components are shown in Equations 3.2 and 3.3.

$$\begin{aligned} member_i &= [gene_1, ..., gene_n] \\ Population &= [member_1, ..., member_i] \end{aligned} \tag{3.2}$$

$$J_i = f_{CostFunction}(member_i) \tag{3.3}$$

Next, the population goes through selection, mating, and mutation that alters the population members based on a defined method; therefore changing the input variables to be evaluated in the cost function. The process will repeat for a defined number of generations or until other stopping criteria it met [29]. The goal is that the members with the optimal cost will survive each generation and the component of continuous random search will avoid premature convergence [35]. GAs have been expanded to applications of parallel processing and handling members with variable length in order to deal with more complex problems [34]. The algorithm components of selection, mating, and mutation have different potential operations – the versions used in the work are discussed below. Figure 3.2 outlines how the process of the GA flows until convergence, or the number of generations is met.

It is also important to note that all of the variables will be generated and altered in binary form. This means the the size of each member is defined by the number of bits allocated for each variable in the member. The number of bits for each gene is determined using the limits of the given variable. Once the population is generated, there are equations to decode and encode the individual genes of each member. These methods and the related equations are described in more detail in Haupt's *Practical Genetic Algorithms* [29]. The difference between having a population of continuous and binary members effects the process of altering the population. The original work

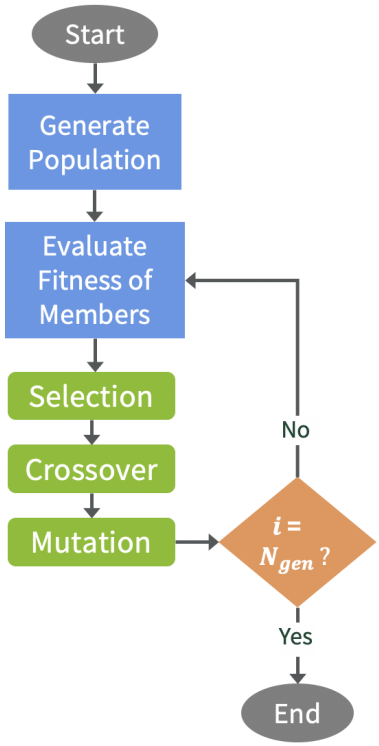

Figure 3.2: Genetic Algorithm Flow Chart

on the STOpS program investigated the differences found between populations of continuous versus binary members and found that a binary GA had a higher success rate of finding the global optimum. For this reason and the effectiveness of the stochastic solver with complex solution spaces, this work will focus on binary GAs.

A simplified population for Branin's function (Equation 3.4) is used to illustrate the process of altering population members in GAs. It is assumed that there is a population of four members each containing two constrained, continuous, and static variables. The mechanisms for selection, mating, and mutation are still applicable to larger populations and more complex cost functions.

$$f(x_1, x_2) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos x_1 + s \tag{3.4}$$

Branin's function differs from Ackley's because it is defined as 2-dimensional with the input variables x_1 and x_2. Equation 3.4 shows the relationship of these two variables along with the constants a, b, c, r, s, and t (values listed in Appendix A) [39]. The three global minima of $f(x^*) = 0.397887$ are shown on the solution space in Figure 3.3. Where x^* is a matrix of the three pairs of x_1 and x_2 solutions. The three solutions to Branin's function are all considered to be a global optimum because they have the same fitness value. This precision of this solutions depends on the number of digits included in pi when evaluating the function. If the same number of digits is used for all three solutions then the optimum will be equal for all three to the desired degree. Note: this problem will be used as a test function for the implemented solvers (Chapter 4.1).

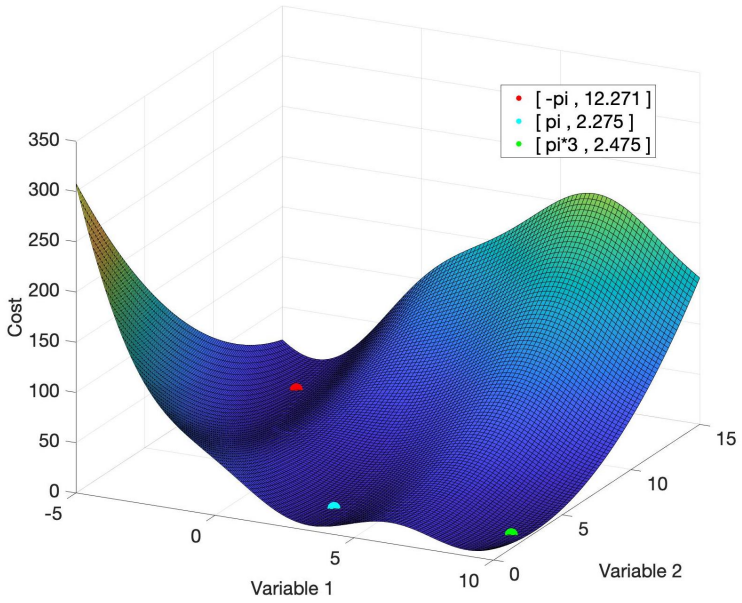

Figure 3.3: Branin's Function

3.2.1 Selection

The selection process is important because it determines which members of the current population to pass onto the new generation. Ideally, if the member with the best

cost is found in an early generation, it will survive mating and be present in the final solution. GAs are not guaranteed to find the fittest member, but judicious parameter tuning can improve the likelihood of doing so. Selection methods sort the population based on fitness value and take N_{keep} of the members from the population for the next generation [29]. The remainder of the next generation ($N_{pop} - N_{keep}$) is filled with offspring that are various combinations of the N_{keep} members from the current population. The offspring are essential because they explore new areas of the solution as they are a new combination of old and new design variables, explained further in the next section (3.2). The value of N_{keep} depends on the problem at hand, a general rule is to keep half of the current population. For large values of N_{keep}, fewer new solutions will be introduced to the population via offspring which may result in premature convergence. When N_{keep} is small there may not be enough parent members to effectively generate offspring [29]. Equation 3.5 shows how N_{keep} is defined by X_{ratio} of the total population.

$$\begin{aligned} N_{keep} &= N_{pop} * X_{ratio} \\ where \quad X_{ratio} &= 0.5 \end{aligned} \tag{3.5}$$

The Branin's function example population of 4 members requires $N_{keep} = 2$.

There are many options for how to select members of the population – this work will focus on the options presented in Chapter 2 of Haupt's *Practical Genetic Algorithms* and is used in the previous version of STOpS [29, 26]. The selected members are used to generate offspring that contain old and new combinations of genes from the parents (Chapter 3.2.2). For the example with Branin's function, Table 3.2 shows the members and their respective fitness (Note: none of the example members contain the known optimal solution).

Table 3.2: Branin's Function Example Population and Cost

Member	x_1					x_2					Cost
1	1	0	0	1	0	1	0	1	1	1	0.6346
2	1	0	0	0	0	1	1	0	1	1	1.635
3	0	0	1	1	0	0	1	1	0	1	0.9121
4	1	1	1	0	1	0	1	0	1	0	2.0841

1. **Random Selection:** This simple method randomly selects the population members to keep based on the value of N_{keep}.This can be executed in a program by generating N_{keep} random numbers and selecting the members with that index in the population. The population in the Branin example would be selected based on two random numbers.

2. **Natural Selection:** This method is also referred to as "survival of the fittest" because it sorts the members from best to worst cost and selects N_{keep} of the best members for mating. This method works well because the best members are always passed onto the next generations. In the case of the Branin example population, no members contain the optimal solution but members one and three would be selected for mating because they have the best fitness (Table 3.2). The next generation would contain these existing members and two offspring that contain old and new combinations of parent genes.

3. **Threshold:** Members are selected based on a "threshold" value that is defined with knowledge of the expected range of solutions. In early generations, there may not be N_{keep} members that meet the threshold value and in this case random random members will be generated to fill the remaining parent slots. If the threshold value for the Branin's function example is set closer to the known optimal solution, e.g. 0.5, then none of these population members would be selected. With a less conservative threshold value, e.g. 1, the N_{keep} quota is met by three of the members and the best two will be selected. A threshold

value of 0.75 is only met by member one so the other N_{keep} spot would be filled with a randomly generated member.

4. **Weighted Random Pairing:** This method assigns a probability to each member and ranks them. The probability is inversely proportional to the cost. So a higher probability means better cost. Next the members are selected based on this probability from a rank system or cost. In the rank system a random number between 0 and 1 is generated and then the members are checked one by one until the probability is equal to the number. When evaluated with cost, the probability is found with a normalized cost value, and then the same process of selecting the embers is used as described above.

3.2.2 Mating

This process is also referred to as "crossover" since it mimics the process of chromosomes evolving and trading genes for the resulting offspring in biology. This stochastic process proves to be efficient in nature and is mimicked in this computational mechanism. Two parent members are selected and combined to form two offspring that result in a combination of old and new parent genes that progress the solution. The number of offspring generated through mating is determined by $N_{pop} - N_{keep}$ and can vary depending on the value of X_{ratio}. This work will use the standard mating methods presented in Chapter 2 of Haupt's *Practical Genetic Algorithms* [29].

1. **Uniform Crossover:** Two members will exchange binary values up to a single, predetermined location and the resulting chromosomes of mixed genes are the offspring. The location of the crossover is randomly generated based on the total length of the member and is in the same location for each parent. This method is considered to be a generalization of crossover methods and extensive

research by Syswerda showed that uniform crossover is almost always more effective at combining members for the new population [29]. Assuming the uniform crossover occurs at bit three, Figure 3.4 shows the resulting offspring from crossing members one and three of the Branin's function example.

Member	x_1					x_2				
1	1	0	0	1	0	1	0	1	1	1
3	1	0	0	0	0	1	1	0	1	1
5	1	0	0	0	0	1	1	0	1	1
6	1	0	0	1	0	1	0	1	1	1

Figure 3.4: Branin's Function Example with Uniform Crossover

The new members five and six are created with different combinations of the two parent members. The blue and gray highlighted strings in Table 3.4 indicate the bits that switch for in the offspring and the non-highlighted bits did not change. In this simple example x_2 does not change in both offspring, so is not benefiting from mating specifically but may prove to be a desired gene as a whole since it belongs to one of the "best" members. On the other hand, x_1 has taken on a new value in both offspring which introduces new areas of the solution space. Depending on the crossover location the offspring may take on completely different variables that help to progress the solution.

2. Random Crossover This method is similar in concept to uniform crossover except the crossover occurs at multiple points in the chromosome. The specific bit locations are selected at random which leads to very different crossover schemes for subsequent pairs. The number of crossover points is constant for

all generations and must consider the total number of bits in the chromosomes to effectively produce offspring. For a chromosome with ten bits, using three crossover points would cause a large change in the offspring because there are only nine potential crossover locations. Larger chromosomes do not necessarily benefit from more crossover points because that may see out desired genes if too many points are used. Figure 2 shows the resulting offspring of the natural selection population for Branin's function with 2 crossover points.

Member	x_1					x_2				
1	1	0	0	1	0	1	0	1	1	1
3	1	0	0	0	0	1	1	0	1	1
7	1	0	0	0	0	1	1	0	1	1
8	1	0	0	1	0	1	0	1	1	1

Figure 3.5: Branin's Function Example with Random Crossover

The crossover occurred at bits three and eight and produced two offspring that have completely new values of x_1 and x_2. The effectiveness of the crossover will be revealed when the new variables are evaluated by the cost function.

After the mating process between two members is complete, the crossover probability, p_c, is considered before adding offspring to the next generation. For each set, a random probability between 0 and 1 is generated which is compared to the defined value of p_c.

3.2.3 Mutation

After the new population is generated, the last alteration is the mutation of individual bits that occurs based on the probability of mutation, p_m. A given population member with length N_{bit} and the probability of mutation, p_m , will result in M bits mutated [29].

$$M = p_m(N_{pop} - 1)N_{bit} \tag{3.6}$$

M randomly selected bits from the population flip their value – 0 to 1 and 1 to 0. This causes a change in the variables that experience mutation which can potentially eliminate a desired gene. The change also introduces new areas of the solution space which can help prevent premature convergence. It is standard for the probability of mutation to be lower to prevent altering the desired members of the population [35].

For example, Branin's function assumes $N_{bit} = 10$, $N_{pop} = 4$, and $p_m = 0.3$ which results in three bits altered from Equation 3.6. Each mutated bit is defined by a row and column position that is randomly generated. For this example the row values are (1 , 3 , 2) and column values are (5 , 1 , 9). The bits that will be mutated are highlighted in Figure **??** and will flip states changing the related variable. After this, the next generation will begin the process again and the effectiveness of the evolutionary mechanism will be revealed when the stopping criteria is met.

3.2.4 Limitations

One of the limitations of GAs is the inability to handle cost functions that require variable length members. This can occur when the number of design variables depends on another so there are both continuous and discrete variables present. This disrupts the uniform population size required for performing crossover and mutation [29].

Member	x_1					x_2				
1	1	0	0	1	0	1	0	1	1	1
2	1	0	0	0	0	1	1	0	1	1
5	0	0	1	0	1	0	1	0	1	0
6	1	1	1	1	0	0	1	1	0	1

Figure 3.6: Branin's Function Example Mutation

Achieving the ability to analyze this type of cost function will present the problem of increased solution space due to the different combinations of continuous and discrete variables.

3.3 Hybrid Optimal Control Problems

The Hybrid Optimal Control Problem (HOCP) evaluates systems with continuous and discrete variables using nested loops [23]. This method separates the variables so that they may be evaluated with a GA or other optimization algorithm as a constant length chromosome. The outer loop will generate members of continuous variables that determine the constraints of the discrete variables optimized in the inner loop. The inner loop considers each member of continuous variables independently in order to evaluate different length sets of discrete variables [23].

In the classic example of the Traveling Salesman Problem (TSP) there is a defined set of Q cities that can be visited with the objective to minimize the total distance traveled between them [23]. Figure 3.7 is an example of a solution to the TSP where all of the cities, $q \in Q$, are shown with the optimal path between them. If the number

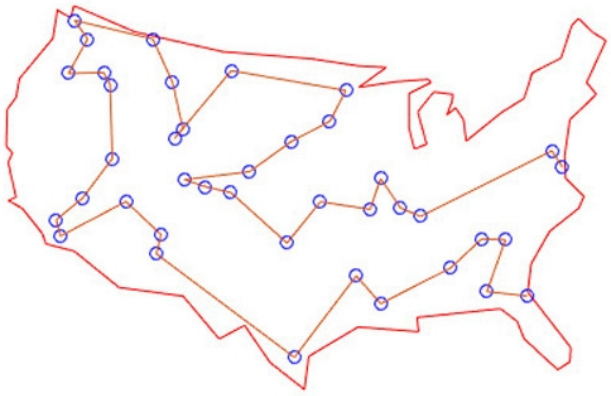

Figure 3.7: Traveling Salesman Problem Example [4]

of cities varies, the number of distances to travel or legs between cities to be evaluated changes. Applying a HOCP, the outer loop will generate a vector **q** of cities $q \in Q$ and the inner loop will optimize the distance between the number of cities defined by **q**. The outer loop that will evaluate the cost of each **q** and use an optimization scheme to find the best **q*** of the generation [23].

Research has found that using GAs as the outer loop solver is more effective than methods like Branch and Bound (B&B) because it explores a more diverse areas of the search space [23]. The B&B method systematically creates a tree structure of all design variable combinations that are evaluated. This method is limited when evaluating cost functions with discrete and continuous variables because it can eliminate sets of continuous variables that actually define the desired solution space. A variety of heuristic algorithms may substitute for the GA in the outer loop, similarly to the inner loop. Research has investigated different EAs to serve as the inner loop solver but the most general one to use is the GA [23]. The inner loop will perform the most function evaluations so it is important that an efficient solver is used.

One limitation of this application is that it requires a large amount of computing power. The inner loop optimization improves the solution of discrete variables for a related continuous set by increasing the number of function evaluations. This method may be better for a problem when there is little known about the desired solution because it is designed to automatically explore a large amount of potential solutions. It separates the problem so that each loop evaluates similar components allowing a more extensive search of possible combinations of continuous and discrete variables.

3.4 Variable Size Design Space Problems

The design space of Ackley's function is defined by the number of variables or dimensions included and when evaluated with an EA, the design space is constant for all generations. For the TSP with a varied number of cities to visit, the subsequent number of input variables changes between members in the population resulting in a variable sized design space (VSDS). This problem cannot be solved with a standard EA because the different length members cause problems for the process of crossover and mutation. The concept of hidden genes can be applied to the functions of a GA in order to solve VSDS problems.

An approach to the VSDS problem is the hidden gene genetic algorithm and is defined by L_{max} or the maximum possible number of design variables [6]. This algorithm mimics the natural process of chromosomes skipping over genes that are not relevant to the type of cell being generated (e.g. chromosome will skip over genes for eyes when generating nose). There are random, probability based, or logical mechanisms available to determine each gene's tag which refers to whether or not the gene is hidden or active. When a gene is hidden it is not evaluated in the cost function, this allows for members of different length to be considered [6]. This work will only focus

on the logical application because it allows more control over which genes are hidden and active relative to the variables involved in the problem.

The only difference between a GA and HGGA is the process of evaluating which genes of each member are hidden and active. In a logical sense there is a marker gene that determines the given chromosome's tags. For example, Branin's equation is 2-dimensional so each chromosome as two genes. Implementing the HGGA would create a chromosome with $L_{max} = 3$ where the first gene is a marker describing second and third gene tags. The value of $gene_1$ would have the limits [0,3] because there are three combinations of possible tag combinations between the two genes. This idea can be expanded to problems like the traveling salesman because the marker genes can be used to serve the discrete variables. Then, the number of discrete variables that are dependent on the continuous parameters would change based using the concept of logical tags [6].

The limitation of the HGGA method compared to the HOCP is that there is not as much iteration of the potential solution of the discrete variables. This means that if the correct marker value is generated, it may not be paired with optimal discrete values. With the Branin example, this would occur every time the second and third genes are both active (since not of the optimal solutions contain a member (x,y) that is equal to zero). This method will require a less expensive run time than a HOCP because there are fewer computations overall.

3.5 Generalized Island Model

The idea of an Island Model optimization is derived from the structure of a GA [34]. This method utilizes the idea of parallel processing when multiple algorithms run in parallel, evaluating the same problem. The GA structure comes in when

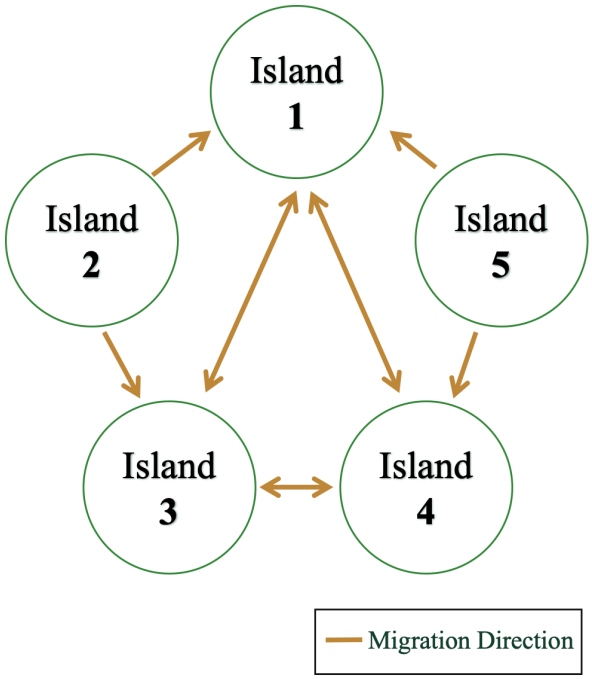

Figure 3.8: Example Island Model Archipelago

these algorithms share population members with each other. This is the same as the process of creating a new population because each algorithm selects members to keep and replace, except in this case the new offspring are filled with shared members.

An example of an Archipelago is shown in Figure 3.8 and describes the given model parameters. The Archipelago is composed of a number of base optimization algorithms or islands that are connected with a defined topology. The topology defines which islands communicate or share solutions with each other. The migration describes how solutions are shared between islands. Overall, the archipelago can be defined as a couple composed of the set of n islands, $\boldsymbol{I}$, and the migration topology, T [34].

$$A = \langle \mathbf{I}, T \rangle \tag{3.7}$$

Each island in $\boldsymbol{I}$ has four components, the optimization algorithm (A), the population (P), the migration selection and replacement policy (S and R). Where the population

is composed of the evaluated fitness value, J, and related population member P_i.

$$I_i = A_i, P_i, T_i, R_i \quad for \quad i = 1, ..., n$$
$$where \quad P = \langle J, P_i \rangle \tag{3.8}$$

Figure 3.9 shows the process a given archipelago follows and at what point solutions are shared based on a defined number of migrations [34]. Similar to GAs, this model also requires tuning of the best parameter values. One of the factors to consider is the number of islands to use because using a large number of islands will not always provide a better solution [34, 26]. Having more islands will require more intricate design of the topology and migration policy. More islands will result in increased run time and this will be experienced more depending on the type of migration. The migration occurs when the migration interval, μ, is met and the stopping criteria is normally a defined number of migrations to occur between islands.

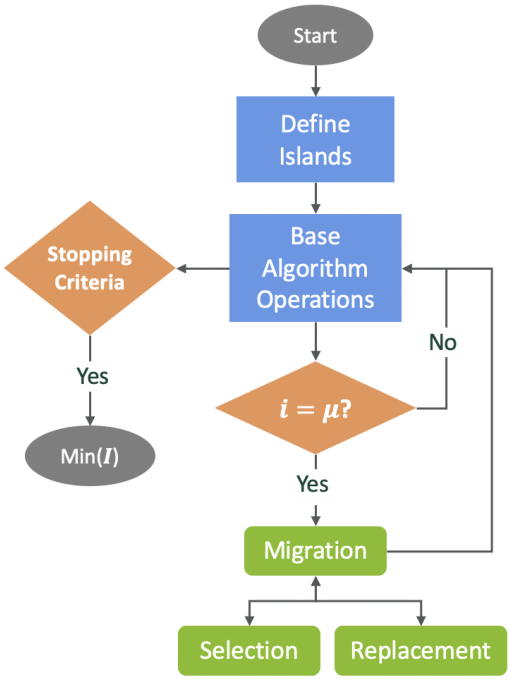

Figure 3.9: Generalized Island Model Flowchart

The migration can be asynchronous or synchronous and describes if solutions are shared between islands at the same time or independent of the state of the other islands. Asynchronous migration has good scalability because it efficiently utilizes available computational resources by not requiring each island to wait for the slowest to finish. This migration benefits models with a large number of islands. While the synchronous migration may result in a slower model, it provides a deterministic process that is easier to monitor and control. There is no evidence that the migration type effects the model performance, only the run time [34].

When there is a defined migration path between two islands the direction of information flow also needs to be specified. It gives the option to prevent some solvers from receiving solutions while still sharing them. For a given island there is a selection policy that picks out members from the population for sharing. Then the island replacement policy describes what members will be removed for the incoming shared members [34]. The number of members that an island replaces can effect its performance. The simplest implementation of this is sorting the population of each island from best to worst, then selecting n members from the top of the list to share with adjacent islands. Figure 3.10 shows an island that is accepting n solutions and replacing m members from the bottom of the sorted population [34].

Initial research into this model used a homogeneous archipelago composed of GAs [34]. However, it is possible to use other evolutionary, heuristic, or local search methods because the different nature of these algorithms can perform better depending problems. A heterogeneous archipelago allows individual islands to benefits from the strengths of adjacent islands and improve performance. The ATC's research into the island model with Pygmo/Pagmo demonstrate that heterogeneous models outperform the corresponding homogeneous one [34]. The amount of migration for the different Archipelagos varied to compare performance for different amounts of shared solutions.

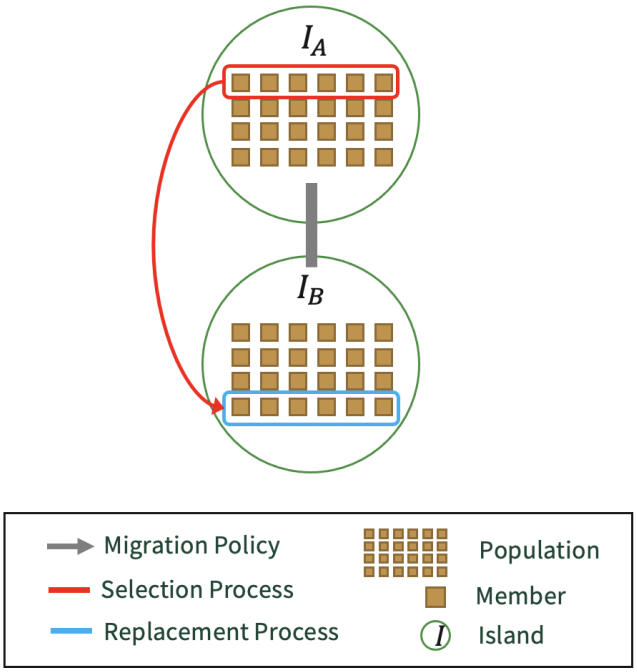

Figure 3.10: Selection and Replacement Processes

The ACT found that in general, migration benefits the model but Fitzgerald notes that too much migration may accidentally eliminate desired solutions [34, 26]. The diverse set of potential topologies was investigated revealing that the base algorithm selected impacts the performance of a topology the most compared to other factors like number of islands and problem. The main takeaway from the characterization of this model is that the performance depends on the pair of cost function and base algorithm [34].

Chapter 4

ALGORITHM VERIFICATION

Two test cases are selected to test the HOCP and HGGA methods: Branin's Function and the Mariner 10 trajectory. The goal is to test to test the processes introduced for evaluating problems with discrete and continuous variables. Testing these functions provides metrics for comparison between the two methods to determine the most effective mechanism for selection and mating, as well as overall feasibility of implementation of the island model.

The HOCP approach allows the base algorithms in both loops to operate with standard genes and can be implemented as a binary GA that is outlined by Haupt in Chapter 2 of *Pratical Gentic Algorithms* [29]. This was done in order to compare the HOCP model to the HGGA for the different GA selection and mating methods. A more extensive investigation to the HOCP with spacecraft trajectory design was presented by Englander where he investigated substituting the GA with other EAs and more complex trajectories [24].

The HGGA implementation uses the same GA from Haupt, but it differs from the HOCP version in the way it generates and evaluates populations. The specifications of the altered population member for each test function will be introduced in the following sections. Other work has investigated the application of the HGGA for MGADSM problems and how more specific mechanisms that deal with the hidden genes effect the performance [6]. This work will use a simple implementation of this algorithm in order to focus on the island model implementation.

These tests were run in MATLAB on a 2.6 GHz 6-Core Intel, 2.2 GHz 2-Core Intel, and 1.4 GHz Intel Core processors. The run time for different tests was only recorded when the same processor was used. All of the Branin runs were completed on the 2.6 GHz 6-Core Intel processor so the execution time provided a useful metric with which to compare the various tests. The Mariner 10 tests were completed on multiple processors. The distributions and statistical other information was generated with the SAS software JMP.

4.1 Branin's Function

Branin's function has three global minima all equal to $f(\mathbf{x_1}, \mathbf{x_2}) = 0.397887$ and can be altered to consider the continuous variable C that determines the values of x_1 and x_2, shown in Equation 4.1.

$$f(C, x_1, x_2) = \begin{cases} C = 0, & 0 \leq x_1 \leq 15, \quad -5 \leq x_2 \leq 10 \\ C = 1, & x_1 = 0, \qquad -5 \leq x_2 \leq 10 \\ C = 2, & 0 \leq x_1 \leq 15, \qquad x_2 = 0 \end{cases} \tag{4.1}$$

The updated limits to Branin's function in Equation 4.1 show how the function changes when either x_1 or x_2 is equal to zero. None of the optimal solutions have x_1 or x_2 equal to zero so ideally, the methods analyzed here will have the number of population members with C = 0 increase over the generations. To investigate the difference in how each method evaluates the continuous variable C, each was run with five selection and two crossover methods, with constant crossover and mutation probabilities.

To compare the consistency of each test method the shape and fit of the distribution of the minimum cost of the final generation are compared. When a distribution is generated, each sample in a set is placed in a bin with a defined range. The number of bins used can change how the spread of the samples appears. Distributions will be compared using the same scale with a small number of sets at a time to avoid skewing the results. The distributions tend to be skewed so the mean is not a particularly useful parameter for comparison. The standard deviation, however, does tend to reflect the differences seen in the results. These metrics can only be used to estimate relationships between the different data sets due to the skewed distributions of sample sets. With test problems that simulate a real world problems, it is important to consider the minimum of each sample set because it shows the best value an algorithm is capable of achieving. In the case of scalable test problems, it is better to look at the sample set as a whole to evaluate the consistency of the solver.

Table 4.1: Generic GA parameters

Npop	100
Ngen	30
keep	50
pc	0.7
pm	0.3
threshold	0.75

To isolate the impact of the selection and crossover methods, a generic set of remaining parameters were selected based on other work [6, 26, 29]. For the population size, 100 is a reasonable size because of the already large amount of function evaluation and was used in other HGGA work [6]. The number of members to keep for the next generation is just half of the total population, the generic setting presented by Haupt [29]. The crossover and mutation probability can be tested to find desired values but this work will use generic values [29, 26]. The threshold value is only used for the threshold selection method and depends on the cost function being evaluated. For

Branin's function a threshold 0.75 that is 0.35 above the solution was used because if the value is too low is may make the solver less effective. For Mariner 10 a value of 17 km/s was used because it is also slightly above the known optimal value of the total delta-V (Chapter 4.2).

4.1.1 HOCP

With the HOCP, the outer loop generates the continuous variable C, that determines the limits of the discrete variables x_1 and x_2, shown in Equation 4.1. Both GAs run with the generic conditions in Table 4.1. It took approximately 18 hours to run all 10 test cases 30 times. A summary of the test cases is shown in Table 4.2 and a more extensive compilation is given in Appendix A. For 10 samples, the mean minimum values were considered with the standard deviation to see how consistent the solver is which is also related to the average value of C.

Table 4.2: HOCP Branin's Function Results

Uniform Crossover				
	Avg Min Cost	Avg Final C	Avg Time (s)	Std
Random	0.4179	1.01	138.4	1.3755
Natural	0.3980	0.45	196.3	0.0716
Threshold	0.3979	0.47	214.9	0.0095
Rank	0.4051	0.59	198.2	1.3697
Cost	0.4232	1.03	226.9	1.8811
Random Crossover				
	Avg Min Cost	Avg Final C	Avg Time (s)	Std
Random	0.4134	0.77	215.2	1.3037
Natural	0.3982	0.46	236.9	0.0319
Threshold	0.3979	0.45	252.3	0.0047
Rank	0.4070	0.57	235.1	1.1987
Cost	0.4186	0.81	236.9	1.7363

All of the selection methods consistently provide a cost that is close to the known solution, but natural selection and threshold perform better than the other three. Distributions of the tests show the trend of how consistent the different methods are. As shown in Table 4.2 the standard deviations which are relatively low, with natural selection and threshold having the lowest values. When the selection method is constant, the crossover methods results in a similar distribution shape showing that the crossover method does no have an impact on performance in this case. As an example, the distributions for natural selection are shown in Figure 4.1.1. The other four are found in Appendix A and will all be considered here. The distributions show the concentration of solutions with the count value on the y-axis in different ranges of solutions along the x-axis.

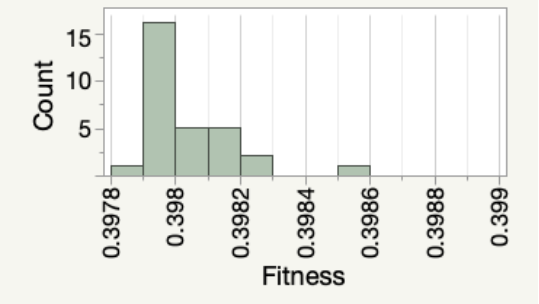

Figure 4.1: HOCP Natural Selection Distributions for Branin's Function with Uniform Crossover

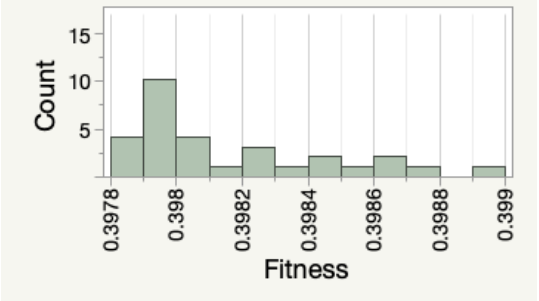

Figure 4.2: HOCP Natural Selection Distributions for Branin's Function with Random Crossover

The nature selection distributions have similar shapes and are skewed to the left. In Figure 4.1.1 the uniform method has a tighter distribution than the random method indicated by the higher density of samples in the tail than on the right side. This shape could be a result of the increased variability in the random method resulting in more outliers than the uniform case. Comparing distributions provides potential relationships between different sample sets. Figure 4.1.1 shows how to peak of both crossover methods for natural selection occur at the same bin range. This indicates that the crossover method does not have a large impact on the overall performance for natural selection. Random, threshold, and rank weighted selection methods also have similar mean values and distribution shapes between the different crossover methods. Again indicating the crossover methods does not have a large impact on overall performance.

The distribution of cost weighted reveals a difference in the applied crossover methods (Appendix A). Random crossover has a tighter distribution with a tall peak while the uniform crossover as a wider spread that is relatively flat – they both have similar means. Even though cost weighted with random crossover has a tight spread, the mean value is higher than the natural selection and threshold methods. Similarly, the random distribution has a tight distribution but a larger mean than natural selection and threshold. The distribution of random with uniform crossover is largely skewed by one outlier. Overall, these distributions show that the real difference in results is due to the selection method with natural selection and threshold outperforming the others.

Since it appears there is no significant difference in the crossover methods, the distributions of Natural Selection and Threshold with uniform crossover are compared in Figure 4.1.1 (random crossover produces a similar result). The distributions are set on the same scale which emphasizes their different shapes. The distributions in

Figure 4.1.1 are both skewed to the left, but the natural selection distribution has a tail, showing that more outliers are present in the sample. With this scale, it also shows that the peak of the threshold distribution is lower than natural selection. One reason for this difference may be that the threshold method is driven by a cutoff value. Where as the natural selection method suffers from occasional premature convergence causing rightward skewing. The spread of the natural selection distribution is more broad than threshold which may benefit certain problems.

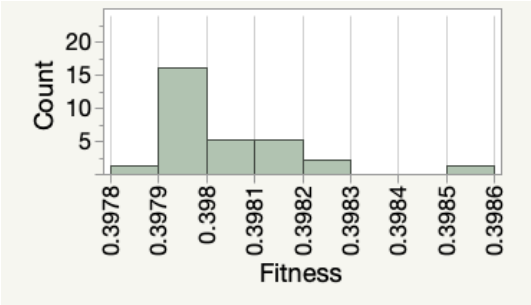

Figure 4.3: HOCP Natural Selection Distributions for Branin's Function

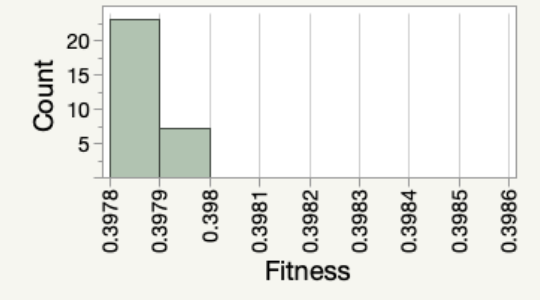

Figure 4.4: HOCP Threshold Distributions for Branin's Function

The generalizations derived from the distributions are supported by the visualization of population trends for each iteration. The average value of C and the final minimum cost of the population of the population are plotted for 30 generations for all 10 test cases in Figure 4.5. These trends reveal some of the benefits of the HOCP set up and the first example of this is the minimum cost. The trend of natural selection and threshold is a straight line the is relatively close to the known minimum value.

These selection methods are effective at finding the optimal solution early on. This is because the inner loop is efficient at finding the best solution for a given value of C. Thus, the HOCP can be run with fewer generations on the outer loop because there is not much change in the solution over the 30 generations. Englander also notes this factor and implemented a secondary stopping criteria that would end the program when the solution stagnated [24]. The three under-performing algorithms have a trend the oscillates around values that are slightly higher than the known solution.

The evolution of C is important, especially with a problem with a known solution. In this case, it is better if C trends towards zero. By the fourth generation natural selection and threshold methods reach a value that is very close to 0 and continues to oscillate slightly. The rank weighted method has the next best trend that starts around one and drops to 0.6 by the fifth generation, but remains there for the rest of the generations. The trend of random and cost weighted vary slightly around one and never trend downwards significantly. The better performing algorithms have a average C trend that starts high and by the fourth generation levels out to a lower value. The methods with a lower average C of the final population have a lower minimum cost solution.

The last factor that was tracked is the run time of this algorithm. As mentioned, obtaining this data required extensive run time. For all of the selection methods the run time of the uniform crossover method was generally shorter than random. This excessive run time is concerning in a preliminary test case because it is a simple evaluation of variables in one equation. When extended the spacecraft trajectories the required time to evaluate all defined and undefined variables will increase quickly.

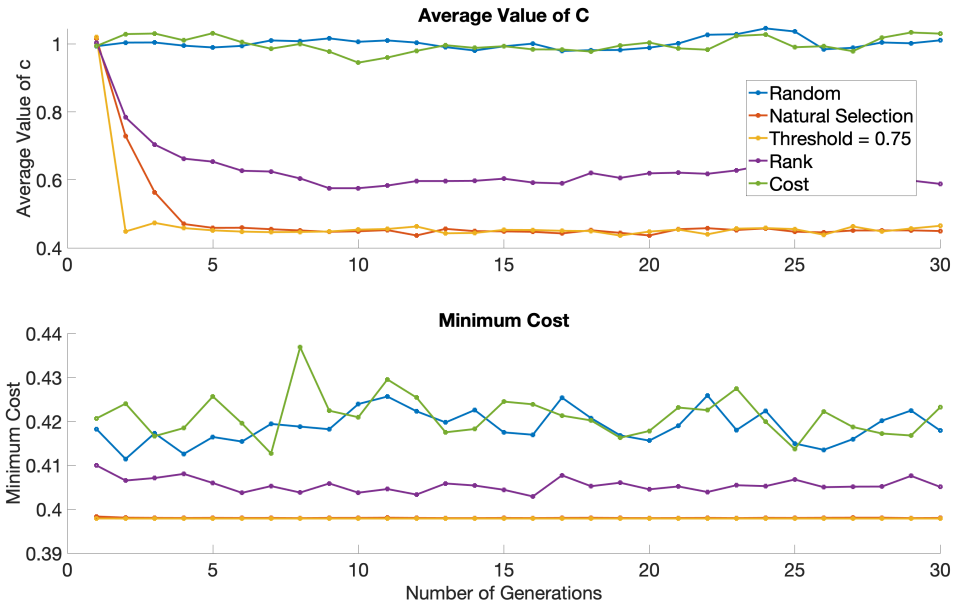

Figure 4.5: Trend of Branin's Function with Uniform Crossover

4.1.2 HGGA

To evaluate Branin's function with a HGGA, each population member is configured to have three genes: C, x_1, and x_2, where C is the marker that determines the range of x_1 and x_2 (Equation 4.1). Table 4.3 shows the hidden and active states of the variables corresponding to C. The generic parameters in Table 4.1 are used with a sample size of 30 and to collect all the data for this test case, it took the HGGA approximately 22 seconds to run.

Table 4.3: HGGA Marker Gene

Marker C	x1	x2
0	active	active
1	hidden	active
2	active	hidden

The restricted capabilities of the HGGA to explore the entire design space of the discrete variable is evident in the results shown in Table 4.4. Only the natural selection

and threshold methods yield a mean that is relatively close to the know solution. The other, more stochastic selection methods do not perform well with this problem. The smallest solution found in the distribution of the results for these poor performing methods are close the the know solution – but the methods do not consistently find that solution.

Table 4.4: HGGA Branin's Function Results

Uniform Crossover				
	Avg Min Cost	Stdev	Avg Final C	Avg Time (ms)
Random	2.111	1.3755	0.1	58.2
Natural	0.4487	0.0716	0	62.7
Threshold	0.4058	0.0095	0	65.1
Rank	01.941	1.3697	0.067	70.1
Cost	2.578	1.8811	0.23	78.1
Random Crossover				
	Avg Min Cost	Stdev	Avg Final C	Avg Time (s)
Random	1.5356	1.3037	0.03	77.9
Natural	0.4289	0.0319	0	72.1
Threshold	0.4030	0.0047	0	83.9
Rank	1.4968	1.1987	0.03	81.3
Cost	2.2715	0.0895	0.03	78.1

When the selection method is constant, the resulting cost is of the same magnitude for different crossover methods. This is seen in Table 4.4, as well as a comparison of the distributions from different methods. Similar to the HOCP analysis, the distributions are compared by crossover method on the same scale.

Overall, the distribution shapes for the different crossover methods are consistent, so again it appears that the crossover method does not effect the consistency of the algorithm performance. While the crossover method may not impact the performance for problems with shorter chromosomes, in more complex cost functions crossover may have an effect. The distributions of random, rank, and cost weighted have a

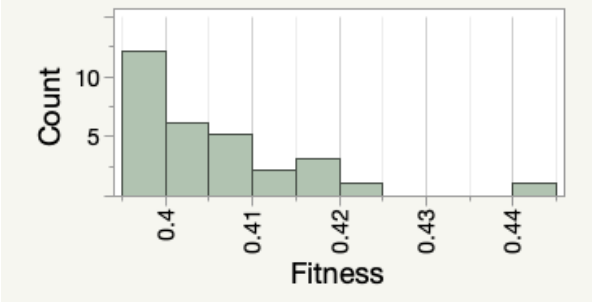

Figure 4.6: HOCP Threshold Distributions for Branin's Function with Uniform Crossover

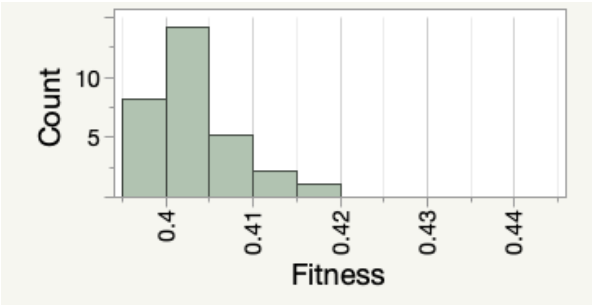

Figure 4.7: HOCP Threshold Distributions for Branin's Function with Random Crossover

wide spread, in contrast to the other two methods with much narrower distributions. The natural selection distributions indicates the most consistent performance though with some outliers. The threshold method is the next best, and while it has a tight fit, it is more skewed to the left with a longer tail than the natural selection. As an example, the distributions for the threshold method are shown in Figure 4.9 – the other four are presented in Appendix A. In the different threshold distributions, the peak occurs in different bin locations. In the uniform method, when the peak is lower, there are more outliers present in the set. This is evidence that the skewed distributions implying that an algorithm is not as consistent. The random method has a distribution that trends more towards normal, suggesting more consistent results.

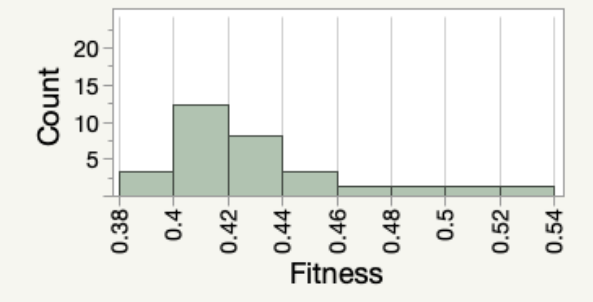

Figure 4.8: HOCP Random Crossover Distributions for Branin's Function with Natural Selection

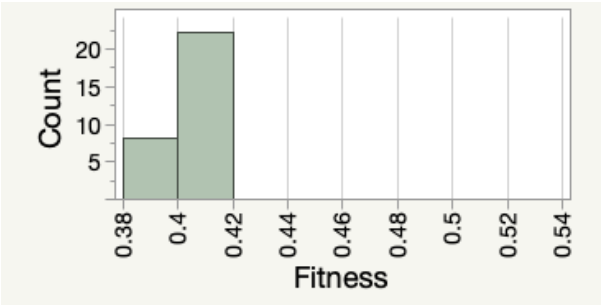

Figure 4.9: HOCP Random Crossover Distributions for Branin's Function with Threshold Selection

Similar to the HOCP analysis, the selection method is the most significant factor affecting consistency. This is seen by comparing the distributions of two selection methods with constant crossover. Figure 4.9 shows the natural selection and threshold distribution for random crossover. The distribution of the threshold method changes when on the same scale as the natural selection. This is due to the fact that the actual range of answers is smaller than natural selection. The skewed nature of this distribution is not shown when the bin size is decreased. The outliers present in the natural selection with random crossover were not apparent when put on the same scale as uniform natural selection because of the difference in range. These two methods have skewed distributions on different scales, but still consistently perform better than the other three methods at finding the best solution.

The trend of different parameters over the number of generations relates to the findings of the distributions. Figure 4.10 shows the trends for the different selection methods with uniform crossover. The two best performing algorithms showed the desired trends for average value of C and minimum cost of the final generation. For the minimum cost, natural selection and threshold started at a higher value and leveled out to a value less than 0.5 around the fifth generation. The other three methods started around the same value as natural selection and threshold but use oscillate around that value for all of the generations. As expected, the natural selection and threshold methods find the optimal value of $C=0$ by the second generation, while the other methods vary slightly from zero throughout the generations.

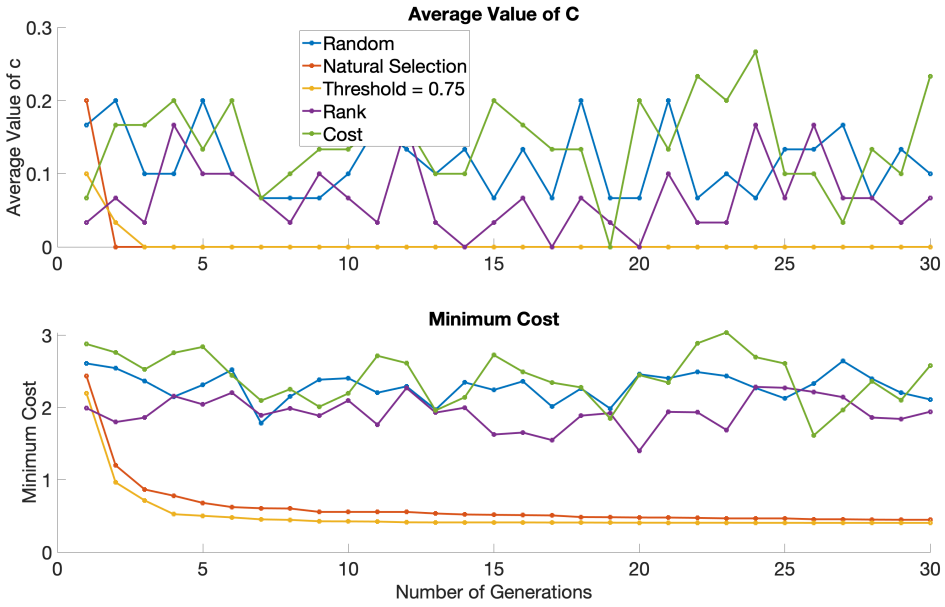

Figure 4.10: Trend of Branin's Function with Uniform Crossover

4.1.3 Comparison

The distributions of the HOCP and HGGA results indicate that direct comparison would not be beneficial because there is a vast difference in the range of the data

sets. This fact does reveal how the smaller and lower range of the HOCP data may suggest the HOCP has a better and more consistent performance. But this should be expected because this method allows the input variable to be optimized with a second solver (the inner loop). Where as in the HGGA, it only has one chance to generate potential input variables. In some cases with the HGGA, the discrete variables generated could be one of the solutions, but if the marker gene is not equal to zero it will miss the opportunity to evaluate the optimal solution. It is interesting that for both algorithms, the natural selection and threshold methods performed best. Figure 4.12 and 4.1.3 shows the distributions of uniform natural selection from the HOCP and HGGA results. They are not shown on the same scale, but to a scale that is automatically created by the JMP software that generates the distribution scale base on sample size and range. Both distributions have a tail to the right but the HOCP is more skewed to the left than HGGA. Skewed distribution suggest that the performance does not have consistent performance.

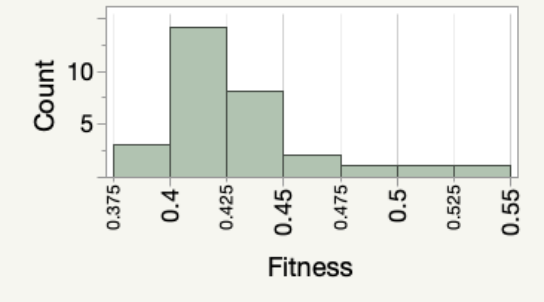

Figure 4.11: Natural Selection for Branin's Function Comparison with HGGA

These tests also reveled the need to consider the run time of these algorithms. The HOCP test cases took 2,947 times longer than the HGGA algorithm. Granted this was collecting multiple samples of different test cases. This is a preliminary test where the goal is to build upon these methods. The individual run time of the HOCP with a simple test function is about 4 minutes, compared too the 50 ms run time of

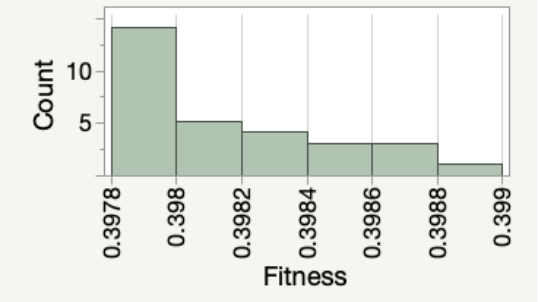

Figure 4.12: Natural Selection for Branin's Function Comparison with HOCP

HGGA. That can be expected in increase with the application of spacecraft trajectory problems and even more so with the island model implementation. While the results of the HOCP appear to be better and more consistent than the HGGA, the run time will cause problems when trying to implement new concepts to the model that might improve the solutions but are computationally expensive.

4.2 Mariner 10

A simplified Mariner 10 problem was used test the performance of these methods for a simple spacecraft trajectory. This real mission was launched on November 3, 1973 from Earth and performed the first planetary fly-by of Venus on February 5, 1974. Due to the trajectory leaving Venus, a large delta-V would be required to enter into orbit around Mercury. The spacecraft instead performed a fly-by of Mercury on March 29, 1974 and then onto a heliocentric trajectory that would allow for two more fly-bys of Mercury and the specific details of the mission are shown in Table 4.5. Using the model presented in Chapter 2, a cost function that takes a planet sequence and corresponding time of flight values to compute the total delta-V of the trajectory. This cost function uses the same method as STOpS so that there is a baseline for comparison.

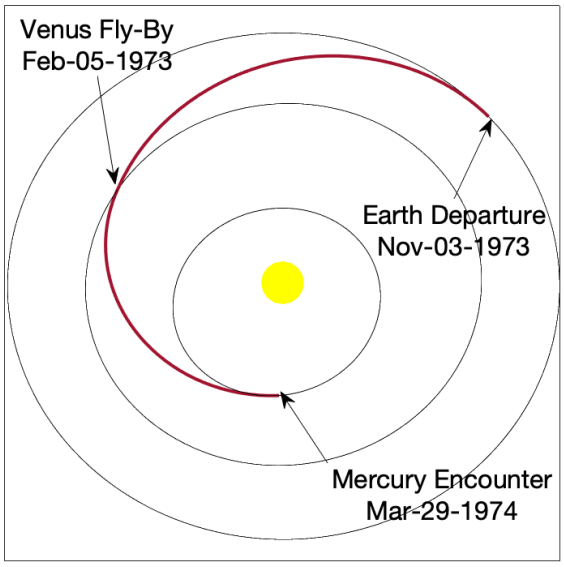

Figure 4.13: Mariner 10 Trajectory

It is difficult to compare the results of this simulation to the actual delta-V used in the mission, this is due to correction maneuvers and actual result of predicted fly-by trajectory. Because of this, the mission dates and Earth-Venus-Mercury (EVM) trajectory are evaluated by the cost function. This produces a delta-V of 16.2485 km/s, which is high because the arrival delta-V is equal to 10.2557 km/s alone. In order to get a better idea of the actual delta-V, the arrival delta-V will be subtracted from the results due to the fact the mission never entered orbit around Mercury resulting in a delta-V of 5.9928 km/s. This solution is not necessarily "optimal" in this design space due to the realities of launch delays, perturbations, and other factors that can impact the trajectory, detailed by Dunne and Burgess in *The Voyage of Mariner 10* [22].

For these tests, there are 30 samples for the HGGA and 10 for the HOCP and these values were chosen based on algorithm run time from Branin's function and past re-

Table 4.5: Mariner 10 Mission Parameters and Variable Limits

	Mariner 10	Lower Bound	Upper Bound
Flyby Planet	Venus	Mercury	Neptune
Departure Date	Nov. 3, 1973	Oct 10,1973	Dec. 1,1973
t_1 (days)	94	70	110
t_2 (days)	52	35	70
t_3 (days)	-	40	100

search [6, 24, 26]. These experiments used the two best performing selection methods from Branin's function tests including both crossover methods to see if crossover has an impact with a larger chromosome size. Initially there were eight test cases between the HGGA and HOCP mechanism combinations. These tests revealed an issue with the cost function that allowed for high turn angle fly-bys of Earth to be selected as the optimal solution. This happened because of two reasons, first the required departure delta-V to enter a heliocentric orbit that has a similar period to Earth is less than a Venus/Mercury bound trajectory. Second, the penalty applied to the fly-by delta-V was not large enough to eliminate these trajectories. For this problem the sum of the departure and fly-by delta-V of an Earth bound heliocentric trajectory values will always be less than just the departure delta-V of a Venus bound trajectory. This causes the solver to generally miss the EVM sequence and converge on the unrealistic Earth-Earth-Earth-Mercury (EEEM) sequence. In the second round of testing the cost function was altered to account for this inconsistency and the tests were rerun with only uniform crossover and both selection methods. Even though the EEEM sequence is not feasible, in these tests it shows how this algorithm performs when solving a problem that has multiple optima.

In the results from Branin's function, when there was a smaller range, or lower standard deviation, the performance of the algorithm was more consistent. This is because there was never a solution with the marker gene C not equal to 0 and knowing the solution to the problem, a good solver would not converge on a different value of C.

In the case of spacecraft trajectory optimization problems, the state of the marker gene has larger impact than in Branin's function besides just changing the number of design variables. For each potential marker gene state, there is a vast amount of specific planet sequences that can be generated. So when looking at a set of solutions from a given algorithm, the statistical parameters must be considered with the different number of planet sequences present in mind.

4.2.1 HOCP

The simple approach to the HOCP used in Branin's equation needs to be extended to handle a variable sequence MGA problem. The outer loop will generate a population of planet sequences that will be evaluated by the inner loop that generates a population of transfer times. In terms of the Branin's function implementation, the variable C will become a vector with length equal to the maximum number of allowed fly-bys. For the Mariner 10 problem there are a maximum of two fly-bys allowed so the outer loop will consider two continuous variables P_1 and P_2 that represent the encounter planets. In order to vary the length of trajectories generated, the idea of *null genes* is implemented. This mechanism is used for decoding chromosomes to be evaluated. Simply, the limits of P_1 and P_2 are greater than the number of planet bodies available and those extra values represent Null values, or no planet encounter. Table 4.6 has the possible values of P_1 and P_2 and their related planet body or Null state. The limits in Table 4.5 will be used in these tests with the exception of the fly-by planet that will increase the limits from [1 8] to [0 15].

Due to the long run time, multiple computers were used to run the tests so the run time was not evaluated but it appeared to be on the same magnitude as the Branin tests. All of these runs resulted in the EEEM sequence as the best solution. The best member for every generation of all ten test runs was also saved and out of the 1200

Table 4.6: HOCP Null Gene Decoding

Continuous Variable	Planetary Representation
1	Mercury
2	Venus
3	Earth
4	Mars
5	Jupiter
6	Saturn
7	Uranus
8	Neptune
9-15	Null

values the EVM sequence only occurred in the first or second generation five times. While the EEEM trajectory is not feasible, it is the global optima for this design space and the HOCP proved to be very good at converging on the global optima in a low number of generations. As discussed with Branin's results, this is because of the second solver that is able to ensure each sequence generated is evaluated thoroughly.

The algorithm consistently produced solutions in the expected range. The mean and standard deviation of these sets become skewed when neglecting the arrival delta-V because some of the solutions found trajectories that a higher departure delta-V but a much lower arrival delta-V, around 0.1 km/s. While the members had a very low cost, they appeared to be large outliers until the individual delta-V components were reconsidered. This shows that subtracting the arrival delta-V is only applicable when the EVM sequence is present so for results with only EEEM the arrival delta-V will still be considered. Some of the EEEM trajectories presented solutions that may have allowed Mariner 10 to enter orbit around Mercury so the solutions should be considered as a whole.

The time variables are expected to be in the same general range because of the tight limits used. The departure dates are all in the lower end of the set limits. The distributions for this data shows some trends but is only composed of 10 samples so

Table 4.7: HOCP Mariner 10 Results

	Avg (km/s)	Std	Min (km/s)	Dep Date UTC	t_1	t_2	t_3
NSU	11.2571	0.0754	11.1566	28-Nov-1973	101	65	67
NSR	8.3796	4.5661	1.6961	29-Nov-1973	81	86	65
TU	9.0278	3.2343	1.5671	27-Nov-1973	107	62	66
TR	11.1778	0.0489	11.1263	25-Nov-1973	99	69	68

Figure 4.14: Contour Plot for Density of Solutions in the Solution Space of Departure Date vs Cost where t_0 is 07-Nov-1973 09:35:59

it is hard to draw conclusions. The standard deviation of uniform natural selection and random threshold were very low, even though these sets did not have the lowest mean or minimum solution. This shows that these models are effective at finding the trajectory sequence of the global minima, but it converges on the inner loop solution prematurely. The random natural selection and uniform threshold had the lowest mean but a high standard deviation showing these models are able to find the best sequence and explore the discrete variable space more.

This is visualized in a density plot of the solutions comparing the departure date and cost. There are three main areas of solutions, the areas that are lower on the

y-axis have desired cost. However, the majority of solutions occupy the area around the higher solution of 11 km/s. Figure 4.14 includes all four combinations but only uniform natural selection and random crossover have distinguishable trends. If the area around 11 km/s was expanded, the details of the density distribution of random natural selection and uniform threshold are shown. Looking at the overall trend of this figure, the departure dates at the lower and upper area of the set limits have a more optimal solution and the random threshold has the most diverse set of solutions.

4.2.2 HGGA

In this problem the population members are constrained by the maximum amount of allowed flybys, in this case 2, resulting in $L_{max} = 7$. The marker gene is the variable n and it will determine the state of other genes in the member. The potential states of a chromosome are shown in table 4.15. The other GA parameters will use the generic setting found in Table 4.1. There did not appear to be an extreme change in the run time for these test cases compared to the results for Branin's function. Unlike the HOCP, these test produced sample sets with different trajectory sequences that makes it difficult to evaluate the set as a whole.

Potential Chromosome States						
n	P_1	P_2	$T_{departure}$	t_1	t_2	t_3
0	P_1	P_2	$T_{departure}$	t_1	t_2	t_3
1	P_1	P_2	$T_{departure}$	t_1	t_2	t_3
2	P_1	P_2	$T_{departure}$	t_1	t_2	t_3
Mariner 10 Mission						
1	V	NaN	1973-11-3 5:43:00 UTC	94	52	NaN

Figure 4.15: HGGA Mariner 10 Potential Chromosome States

The two trajectory sequences that resulted were EVM and EEEM and looking at the whole set the solutions with the EEEM sequence had the best cost. All of the trajectories with the Earth fly-bys were the result of the a large turn angle and none of them found trajectories with a low arrival delta-V as seen in Chapter 4.2.1. These results can be compared with the assumption that the spacecraft is capable of executing the fly-by with an impulsive maneuver and the arrival delta-V is subtracted from all samples. This reveals how the constraints on a spacecraft trajectory optimization problems is important, even more so when there is not a defined sequence. In this case when there were less constraints on one design variable it opened up a new area of the design space with a different sequence.

Table 4.8: HGGA Mariner 10 Results

	Avg	Std	EVM	EEEM
Natural Selection, Uniform	3.5098	1.3092	17	13
Natural Selection, Random	4.1414	0.8245	25	5
Threshold, Uniform	4.2952	0.7057	25	5
Threshold, Random	4.0347	0.8381	23	7

The ratio of resulting planet sequences, seen in Figure 4.16, impacts the distribution of the given set. For both threshold types and random natural selection, the EEEM sequence appeared less than 25% of the time, where as uniform natural selection was almost evenly split. All of these distributions are skewed to the right, opposite of the trends seen in Branin's results (Appendix A). The distribution peaks are all centered around the solutions for the EVM sequence and decay towards the lower cost EEEM solutions. In terms of the design space here, the EEEM sequence is the global optimum and EVM is the local optima. Ideally, an algorithm will find the global optima the majority of the time resulting in a low standard deviation and normally distributed solutions. In this case, a lower standard deviation is not necessarily desirable because the samples with a lower standard deviation did not find

the global optimum sequence as often. These test cases show that uniform natural selection is more effective at searching areas of a diverse design space and selecting optimal members from them.

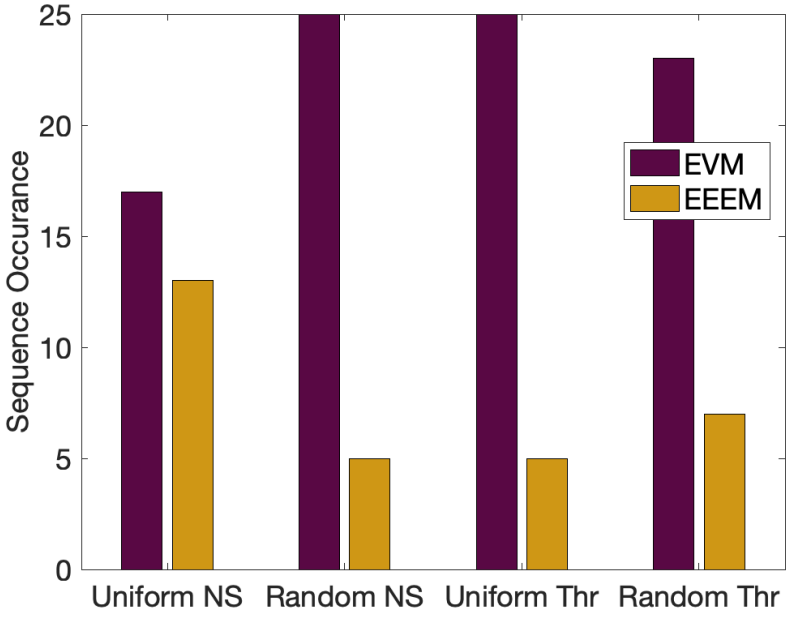

Figure 4.16: HGGA Mariner 10 Sequence Breakdown

EVM Best Solution						
1	V	P_2	03-Nov-1973 22:16:19	90	57	t_3
EEEM Best Solution						
2	E	E	30-Nov-1973 11:32:38	104	67	58

Figure 4.17: Best HGGA solutions for Mariner 10 with Uniform Natural Selection

Looking more closely at the uniform natural selection set, the design variables of the best solution for each planet sequence is found in Figure 4.17 The distributions for each sequence were generated but did not reveal anything because the difference in shape is thought to be the result of difference in sample size. Figure 4.18 shows

the two trajectories for the best solution and it illustrates the expensive maneuver required by the EEEM sequence. The EVM solution was so close to the known parameters of Mariner 10 that when plotted together the lines virtually overlapped. It is obviously not expected for the EEEM solution to have similar transfer times, but the numbers are close due to the fact that the bounds were tight in favor Of the Mariner 10 trajectory.

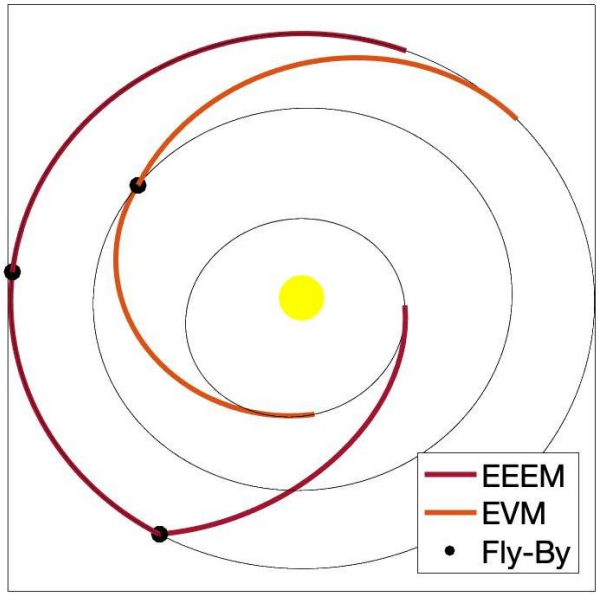

Figure 4.18: Mariner 10 Solution Trajectories for HGGA Sequences

Table 4.9: HGGA Mariner 10 Best Solutions for EVM and EEEM Delta-V

	Total Delta-V	Departure	Fly-By 1	Fly-By 2	Arrival
EVM	4.3742	4.3468	0.0274	NaN	10.2401
EEEM	0.9948	0.0175	0.0017	0.9757	10.6473

The delta-V breakdown is outlined for the best found solutions for each sequence in Table 4.2.2 (both produced with the uniform natural selection method). The arrival delta-V is not considered for this problem and even if it were, the EEEM sequence would still have a lower total cost. The difference in departure delta-V is expected because of the target planet for each sequence. The fly-by delta-V for the EMV

solution was on the higher end due to the difference in dates from known parameters. The second fly-by delta-V for the EEEM sequence exceeds non-powered fly-by limits but it is the lowest recorded value for any EEEM member.

The density plot in Figure 4.19 is more diverse than the HOCP results because of the difference in sample sizes. All of the test cases populated the global optimum area (EEEM) in different capacities and the uniform natural selection solutions occupied the largest amount with subsequent methods following the trends of the mean and standard deviation values. The majority of the solutions are indicated by the darker area and occupies the area around 4-5 km/s across the lower end of the departure date limits. The EEEM solutions here only occupy one of the known solution spaces for EEEM which is around the same cost value with a lower departure date (Figure 4.14).

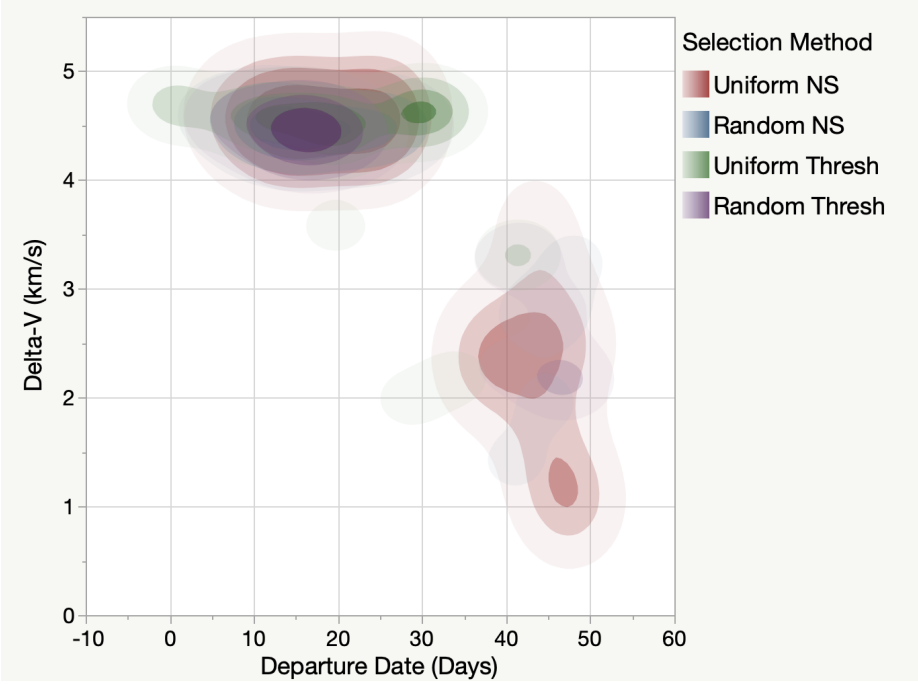

Figure 4.19: Density Plot of All Solution Sequences for Mariner 10 with HGGA where t_0 is 13-Oct-1973 16:36:28

4.2.3 Constrained Mariner 10

In order to evaluate a more realistic spacecraft trajectory problem, the STOpS cost function was altered to apply a larger penalty for the fly-by turn angle. Based on the results from the first test, uniform natural selection and threshold showed a consistent performance for both algorithms. The second Mariner 10 test will use these selection methods in order to compare the performance of HGGA and HOCP with the corrected cost function. These comparisons are made with the fact that the algorithms test sets have different sample sizes in mind. The generic GA settings were used from Table 4.1.

All of the final solutions found the EVM trajectory sequence and this was expected because of the changes made to the cost function. Using this cost function to evaluate the reported transfer times for Mariner 10 resulted in a cost of 5.9928 km/s. Results from STOpS and this work has produced values less than this cost and these solutions have very similar transfer times but do not suffer from the realities that may constrain a mission. The reported Mariner 10 mission having less optimal solution shows why it is difficult to use real missions as a test case – the cost function does not include anomalies that may occur at any step of a mission. The best found solution for each sequence is outlined in Table 4.10.

Table 4.10: Best Solutions for EVM and HGGA and HOCP

Alg	Sel	Avg (km/s)	StDev	Min (km/s)	Dep (JD)	t1 (d)	t2 (d)
HGGA	NS	4.5770	0.0960	4.4235	30-Oct-1973	93	58
	T	4.4993	0.0781	4.4031	01-Nov-1973	92	57
HOCP	NS	4.4231	0.0196	4.3953	06-Nov-1973	87	56
	T	4.4118	0.0161	4.3751	04-Nov-1973	89	57

As seen in all of the previous tests that resulted in the EVM sequence, the time variables are very close to the actual mission dates and when these trajectories are

plotted they basically overlap one another. HOCP produced the lowest final cost, and the small samples also had the lowest means. The HGGA values are relatively close to the HOCP solutions, and only HGGA natural selection did not produce a mean and minimum value that was less than the reported value being compared. All of these sets have low standard deviations, but the difference in the HGGA and HOCP sets is visualized in Figure 4.20. Where the density areas of the HOCP sets are very concentrated in one area compared to the large area that the HGGA sets occupy. For both algorithms, the threshold method performed better than natural selection. But both selection methods are useful depending on the desired results. Natural selection explores a wider area of the design space and does a relatively good job at finding low cost at the same time. While threshold is driven by a specific value so it is confined to a smaller area of the design space seen in Figure 4.20.

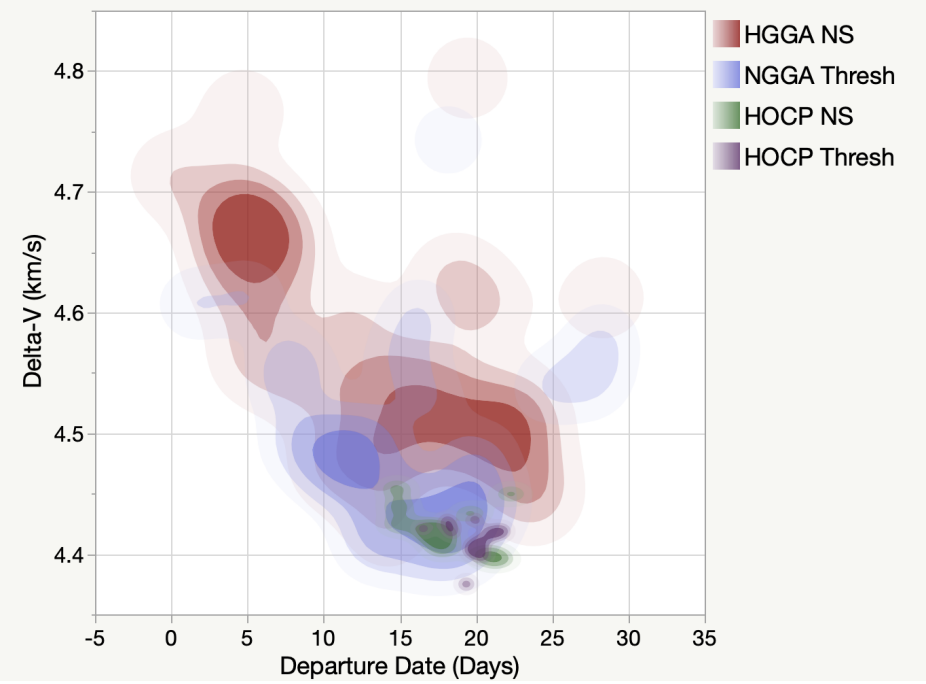

Figure 4.20: Density of Solutions for HGGA and HOCP when t_0 is 16-Oct-1973 05:03:50

The difference in performance of the HGGA and HOCP had been consistent through all these test cases. With the right selection method, both algorithms demonstrate the ability to handle problems with both continuous and discrete variables. While the HOCP is more consistent than the HGGA, it has an extremely long run time that would only increase while any kind of island model implementation. The HOCP already works as a kind of collaborative algorithm system that is very good at finding the global minima. For normal EAs, it has been extensively researched how homogeneous and heterogeneous collaboration and the addition of local search solvers can improve the performance of the algorithms as a whole. It is very likely that if these alterations were made, the result of the inner loop would improve as seen in other research, along with an increase in run time. Acknowledging this, the HGGA is a better candidate for island model implementation because it has a very short run time and is still able to produce solutions in the range of what HOCP achieves. Collaboration of the HGGA requires an algorithm that is also able to handle a VSDS problem.

Chapter 5

ISLAND MODEL FOR A VARIABLE SIZED DESIGN SPACE

5.1 Cassini 1

ESA ACT's Global Trajectory Optimization Problems (GTOP) Database provides open source code that serves as the framework for other research to apply and compare derivative-free solvers for MGA and MGADSM problems [34, 8, 33]. This framework is commonly found in spacecraft trajectory optimization research, most notably in Abdelkhlik's hidden gene work [6, 5]. He mainly focused on the DSM problem using different hidden gene mechanisms, similar to the different selection methods in Section 3.2.1, to identify the effect on performance. This work aims to determine if the benefits of the island model for normal EAs are realized when using a HGGA island model system. Cassini 1 is a MGA problem from GTOP that is similar to the actual Cassini mission without the DSMs and it aims to minimize total delta-V based on transfer time design variables. The problem is designed to evaluate the sequence of Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS) shown in Figure 5.1. The limits of each design variable is outline in Table 5.1. The lowest delta-V of 4.9340 km/s was achieved by Pinna and Izzo using an EA from Pagmo. Abdelkhlik reported a value of 11.2259 km/s using a HGGA, but this was only a preliminary study of the algorithm as of the work focused on DSM problems.

As seen in STOpS, optimization schemes function more effectively when they are thoughtfully designed. This usually requires more preliminary testing of the set-up of an algorithm, but ultimately results in a better solution. In the case of STOpS, all of the EAs were tested individually for Mariner 10. Then sets of islands were constructed

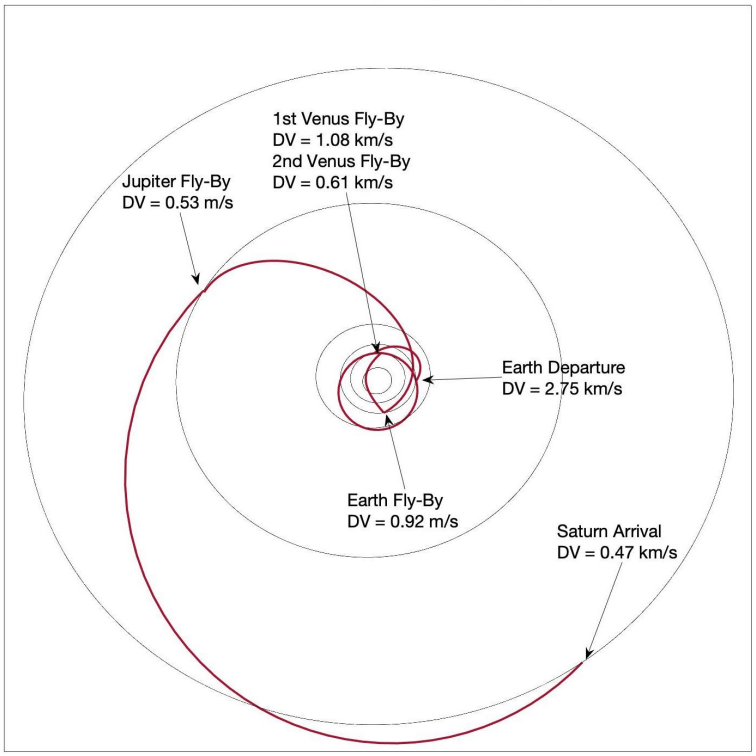

Figure 5.1: Trajectory of the Best Known Solution for Cassini1 from Izzo and Pinna

with the result that the best solution came from the model with the algorithms that performed best in the preliminary evaluation. This work aims to tune the algorithms in a similar fashion, except in this case it is limited to the algorithm parameters. While it was found that in heterogeneous models, the base algorithm parameters do not have large impact on the performance, Izzo did not say whether or not that was the case for homogeneous models. The Branin runs described in Table 4.2 and 4.4 show that natural selection and threshold are the best selection methods. The Mariner 10 problem confirms this and shows a slight advantage towards threshold. The crossover methods do not appear to have a large impact. Abdelkhlik and other researchers in general generally skip to the MGADSM problem because that provides a more realistic trajectory. This work uses the MGA problem to learn about the

consistency of the algorithm performance and how the solution space changes when the island model parameters or problem constraints are altered.

The *mga.m* cost function from the GTOP database is used for these tests [41]. Similar to the function used in Mariner 10 tests, this uses a 2-body patched conics approximation to solve the interplanetary trajectory legs. It assumes that all maneuvers are instantaneous and generates local information on required planet ephemeris. A version of Izzo's Lambert's solver is used in STOpS and these Mariner 10 tests. The *mga.m* function provides an updated version of this solver that uses a new graph method that converges in fewer iterations. This function is designed for powered fly-by maneuvers and this requires the delta-V at the fly-by pericenter radius to be calculated relative to the encounter body. The calculations of the orbits model is described in more detail in Curtis's *Orbital Mechanics for Engineering Students* and other works [34, 24, 6].

The Cassini 1 problem presented by the ACT describes the objective as completing an EVVEJS trajectory to arrive at Saturn with a pericenter radius of 108,950 km and an eccentricity of 0.98. Izzo's best known solution to this problem is shown in Figure 5.1. The limits on the design variables provided by the ACT are very wide, because it is expected this problem will be solved with only six time of flight variables. The limits on encounter planets $P_1, ..., P_n$ for the Mariner 10 tests include all planets in the solar system because the maximum number of fly-bys was low. In this case, fly-by limits are [0 4] because the cost function being used is only designed to handle up to four fly-bys. Figure 5.2 shows all of the design variables along with an example of the hidden genes activated by the marker gene $n = 2$. Below the potential chromosome states for the hidden gene system in Figure 5.2 is the best solution presented by Izzo for reference to the solutions generated with these models.

Potential Chromosome States										
n	P_1	P_2	P_3	P_4	T_0	t_1	t_2	t_3	t_4	t_5
2	P_1	P_2	P_3	P_4	T_0	t_1	t_2	t_3	t_4	t_5
Izzo's Best Solution										
4	2	2	3	5	17-Sep-1856	159	449	55	1024	4553

Figure 5.2: Example Chromosomes with Hidden Genes

The variety of solutions for this problem is much larger than seen in Mariner 10 tests. It was very clear from early tests that this problem requires more constraints depending on the desired outcome. If the GTOP limits are used with the addition of the number planet encounters ranging from [0 4] and encounter bodies ranging from [2 5], the most common sequence is Earth-Earth-Earth-Earth-Saturn (EEEES). The lowest recorded cost for this trajectory is 3.1537 km/s. This is similar to the problem found in the Mariner 10 testing, except in this case the cost function is designed to calculate the delta-V required to execute the maneuver. With the Mariner 10 problem, these Earth fly-by's disappear from the results when the turn-angle penalty is adjusted for non-powered fly-by. The Cassini 1 tests will not alter the GTOP cost function in any way in order to allow for comparison between results that used this model. The design variable limits are changed in order to explore broader areas of the solution space. The two options are to constrain the encounter body, which is exactly what past research has done, or constrain the transfer time. The altered transfer times for Cassini 1 are compared to the GTOP in Table 5.1 and provide a more diverse set of sequence solutions with no constraints on encounter bodies.

Similar to individual EAs, the island model performance can depend on the tuning of the parameters. The island model parameters very similar to the general functioning of a GA (Section 3.5). In the ACT's extensive investigation into the Island Model

Table 5.1: Cassini 1 Variable Limits for the GTOP Version and Island Model Testing

	GTOP		Set 1		Set 2	
n	4		2	4	2	4
P_1	V		V		V	J
P_2	V		V		V	J
P_3	E		V	J	V	J
P_4	J		V	J	V	J
T_0	-1000	0	-1000	0	-1000	-500
t_1	30	400	30	400	100	200
t_2	100	470	100	470	400	470
t_3	30	400	30	400	30	100
t_4	400	2000	400	2000	900	1100
t_5	1000	6000	1000	6000	4000	5000

performance, they used scalable test functions (MGA and MGADSM) quantify the benefits that migration and topology consideration can provide. The one parameter not included in this work is the collaboration of different base algorithms. While other works have developed altered EAs with a hidden gene component, it is not necessary to implement them here because it is well known that collaboration of different base algorithms will improve the result of many optimization problems [6, 5].

Mariner 10 testing demonstrated that threshold is better at finding a lower cost while natural selection is better at searching areas closer to the known optima. With this in mind three different combinations of selection are used for testing the island model. The first two are homogeneous models of all natural selection or threshold where there is two way communication between all connected islands. The third is a combination of natural selection and threshold islands, where the communication only occurs from the threshold island sending solutions to a natural selection island.

5.2 Topology

The ACT's research found that the performance of different migration topologies was most affected by changing the base optimization algorithms. For these tests all of the base algorithms are HGGA so it is not expected that different topology styles (chain, ring, grid, etc.) will impact performance. The ACT also found that increasing the number of islands in a topology does not necessarily improve the performance, and can be restricted by computational platform. Including more islands in a HGGA island model will theoretically allow the solver to explore more areas of the solution space, something the HGGA struggled with in previous testing (Chapter 4). Three Archipelagos with different numbers of islands are tested with the "Set 1" limits in Table 5.1 using the different selection method combinations. These limits constrain P_1 and P_2 to Venus and use the broad time ranges from the original GTOP problem.

The three topologies studied consist of one, six, and ten islands. The multi-island Archipelagos are set in a chain topology shown in Figure 5.3 rather than something like fully connected because more communication does not necessarily result in a better solution. The ring topology provides a standard level of communication between all islands. The ten island model on the left shows the two way communication between adjacent islands. The heterogeneous model on the right shows how the threshold islands send solutions to adjacent natural selection islands. The single island model shares solutions with itself.

The distributions for these sample sets are non-normal and have different variance. Surprisingly, the distributions of the single island model appear to be the most normal, and as the number of islands increase any trends in the distribution decrease (Appendix B). In contrast, the final minimum cost of the model improves when more islands are used. Table 5.2 shows how the results improve when the number of se-

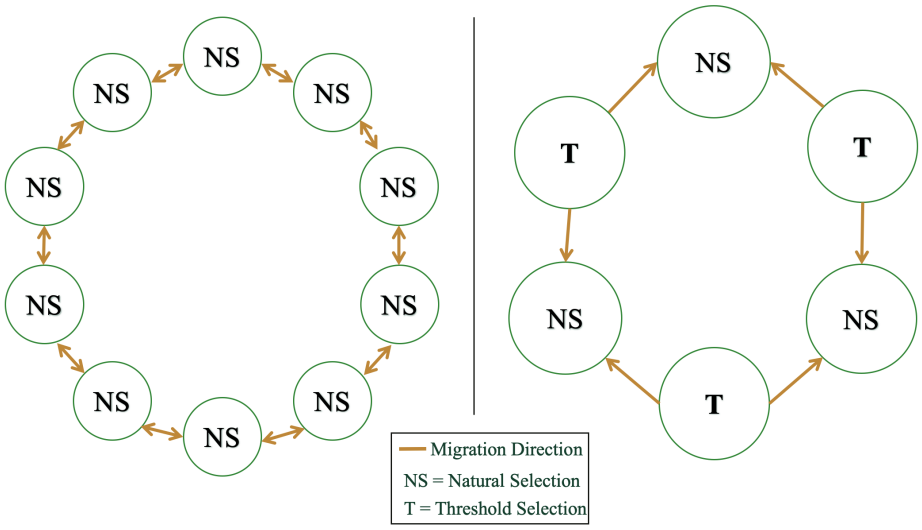

Figure 5.3: Example of 10-Island all NS (left) and 6-Island NS/T Combination (right) Topologies

quences present in the solution set is lower. This implies that for the HGGA, using more islands helps the solver find the most optimal sequences and more effectively explore the discrete variables for a given sequence.

Table 5.2: Results for Different Number of Islands and Selection Methods

Num Islands	Selection	Min ($\frac{km}{s}$)	Avg ($\frac{km}{s}$)	Std	Num Seq
1	NS	14.1	22.4	3.81	8
	T	19.4	62.7	20.1	14
	NS-T	11.1	21.8	3.95	10
6	NS	10.5	18.3	2.07	7
	T	20.0	39.3	11.2	12
	NS-T	8.01	20.8	3.40	9
10	NS	5.65	9.62	1.89	2
	T	5.48	21.2	11.2	9
	NS-T	5.60	10.6	3.03	3

The threshold method consistently has the highest mean and standard deviation but with ten islands found the lowest cost of all the tests. It makes sense that this method would find the lowest cost because as the mean decreases, the threshold process still produces a high standard deviation. So when a larger number of islands is used the

solver should produce a value close to the global optima when a sufficient amount of runs are conducted. The natural selection method was improved by combination of threshold for the tested number of islands. The all-natural-selection model has a larger decrease in standard deviation than that of the combined method as more islands are added. The number of planet sequences of the combined method is greatly decreased with ten islands.

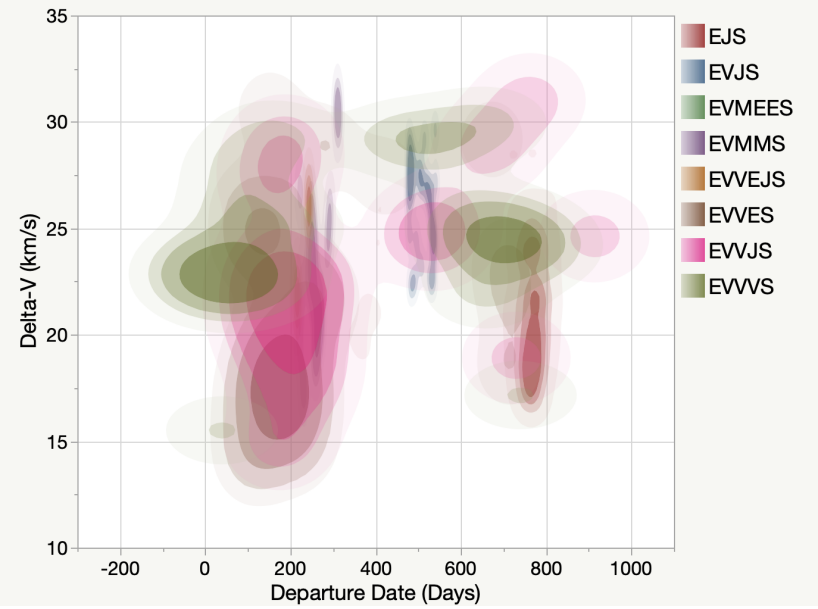

Figure 5.4: Density of Solutions for Natural Selection Models with 10 Islands where t_0 is 19-Apr-1856 13:35:02

Figure 5.4 and 5.5 shows the difference in density of solutions for the natural selection model with one and ten islands. As seen in Table 5.2, the number of sequences found in the solutions decreases as islands are added. With more islands the solver is able to converge on the best sequences. For initial mission planning however, it may be beneficial to also get the results of the one island test in order to see other potential sequences. But this wouldn't necessarily be presenting a more optimal path, because the most common sequence with one island, Earth-Venus-Venus-Venus-Saturn (EVVVS), is not found in any solutions from the ten island tests. The solutions

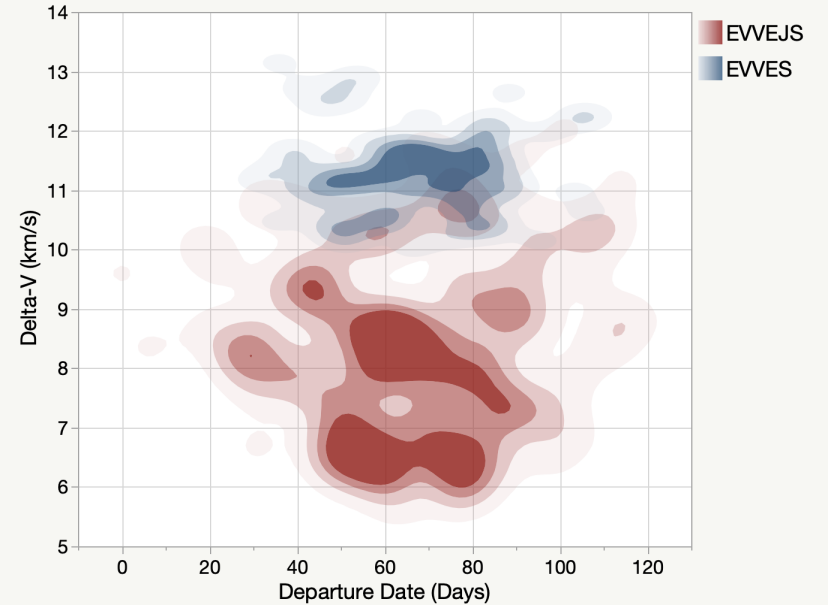

Figure 5.5: Density of Solutions for Natural Selection Models with 10 Islands where t_0 is 19-Jul-1856 14:16:48

for the EVVEJS sequence has cost values from 18-26 km/s for the one island test which decreases with 10 islands. In some cases it may be desirable to be presented with more planet sequences for early mission design phases. On the other hand, getting the more concentrated results ensures that those solutions will be more optimal compared to everything else the solver has evaluated. The settings of the HGGA can be altered to change the number of sequences and cost value. Larger number of sequences results in larger cost values and lower number of sequences with lower cost values.

5.3 Migration

The same ACT research determined that models benefit from migration of solutions by comparing tests of models with and without migration [34]. STOpS tested how more specific values of migration changed the performance and found that too much migration may actually cause the solver to eliminate an optimal solution [26]. With

these findings in mind, a two island model was used to test three different values of migration. These tests use the "Set 1" limits form Table 5.1 where the time ranges are broad and the first two planets are constrained to Venus.

Table 5.3: Results for Different Number of Migrations

Num Migrations	Selection	Min ($\frac{km}{s}$)	Avg ($\frac{km}{s}$)	Std	Num Seq
2	NS	9.43	20.0	2.64	10
	T	19.2	52.3	15.5	22
	NS-T	13.2	22.0	3.29	10
4	NS	5.95	11.3	2.28	2
	T	5.75	34.6	14.8	8
	NS-T	5.73	12.1	2.88	2
8	NS	9.13	17.4	2.12	9
	T	18.6	52.3	15.5	17
	NS-T	10.2	19.1	2.33	9

All of the distributions are still non-normal, but the distributions of samples with less migration appear to be less skewed. This does not correlate to lower cost values. There is a large improvement in solutions when the number of migrations is doubled. However, when the migrations are increased further, the solutions actually got worse as the solver is not able to converge on less sequences. The continuous variables benefit from more migration of solutions, but gain too much migration introduces new sequences that may lead the solver away from a more optimal search area. Natural selection is the most consistent solver for different migration settings, but when the settings are ideal the addition of threshold produces the best cost of these tests. The threshold model is most sensitive, but has shown the potential to have the best cost when the correct settings are used. The standard deviation is the one parameter that didn't change with different migrations. This suggests that the average of a given set is lower if there are less sequences present.

The minimum cost does not improve as much from altering the migration as compared to the number of islands. Similar to those tests, Figure 5.6 shows how the variety of

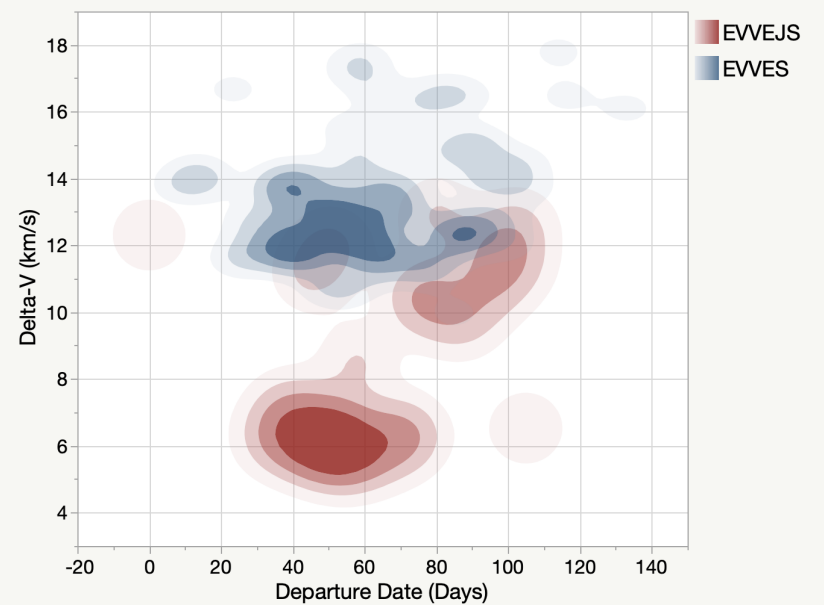

Figure 5.6: Density of Solutions for Natural Selection and Threshold Model with Two Migrations where t_0 is 11-Mar-1856 07:37:55

sequences decreases as the solution improves compared to Figure 5.7 which contains 10 sequence. While changing the migration may not provide the best known solution, it is computationally less expensive compared the adding more islands. The change in migration introduces the solver to new sequences but does not provide the additional cost evaluations that another island provides. A combination of the two mechanisms can be altered depending on the desired results: more sequences or better cost.

5.4 Design Variable Limits

With knowledge of how topology and migration impacts performance, these tests will combine the parameters to evaluate Cassini 1 with the "Set 2" limits from Table 5.1. The limits on the time variables are reduced towards the solution presented by Pinna and Izzo found in Table 5.2. The planets are constrained to Venus to Jupiter which allows for four potential encounter bodies along with the four potential fly-bys. As

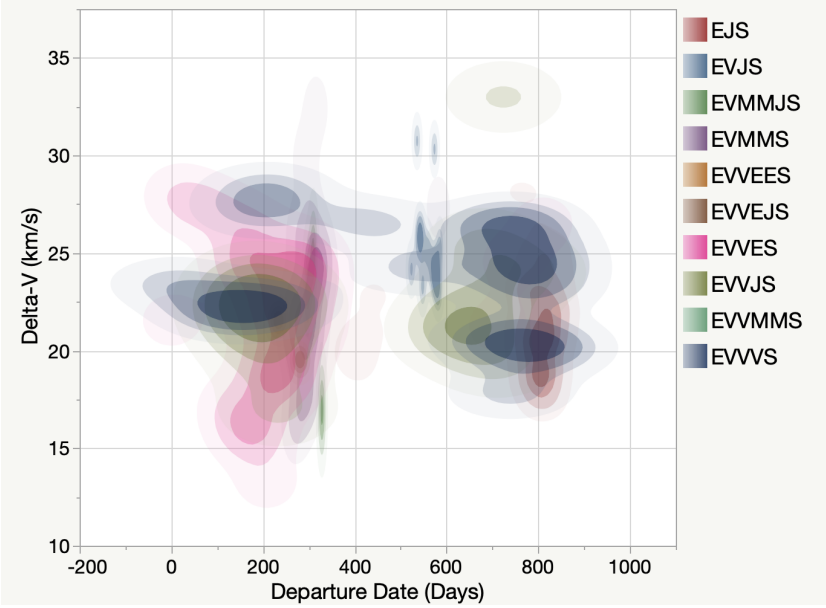

Figure 5.7: Density of Solutions for Natural Selection and Threshold Model with Four Migrations where t_0 is 22-Jul-1856 01:58:04

mentioned, leaving planet variables unconstrained will result in a larger occurrence of Earth fly-bys. Previous testing showed that natural selection has the most consistent performance for a variety of settings but with the right tuning threshold produces solutions with a better cost. Two migration policies for six islands will be compared and utilizing the combination of the natural selection and threshold methods. The third Archipelago included is composed of three islands and all are shown in Figure 5.8. The minimum cost of the model is saved after two, four, and eight migrations and all of the distributions of the sample sets are present in Appendix C.

Expanding the design variable limits for this problem increases the resulting cost of the solutions. This highlights the limitation of the HGGA where as the number of potential continuous variable sequences increases the ability of the solver to converge on the optimal set of discrete variables decreases. The number of migrations and different topologies did not appear to have an impact on the number of resulting sequences. In previous tests some of the models resulted in double the amount of

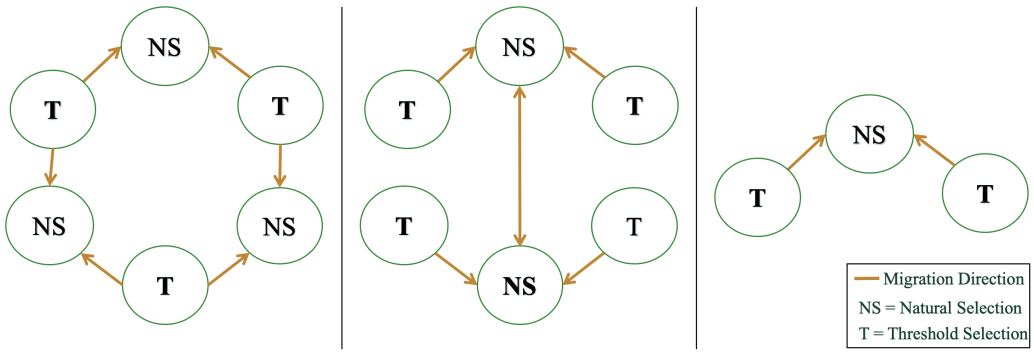

Figure 5.8: Archipelagos for Design Variable Testing

sequences presented here. The standard deviation also experienced little change from the different configurations but the higher values corresponded to a lower overall minimum cost. The higher standard deviation means that there are more values further away from the mean, and in the case of this problem there are outliers to the left of the mean or towards a lower cost. Compared to the other parameters the minimum of the set changes for different numbers of migrations but there is no notable pattern. The lack of characteristics in this data is related to the change in constraints on the first two planet variables, resulting in 256 potential sequences of the four continuous variables. Even when the algorithm collaborates with others it is unable to converge on an optimal solution of the discrete variables due to the large amount of sequences being evaluated. The best solution found for each model is Earth-Venus-Venus-Earth-Saturn (EVVES) and the "3 Islands" model found the lowest cost of all with the trajectory shown in Figure 5.9.

The "6 Islands V1" and "3 Islands" models were run in previous tests with the constrained planet limits and these results allowed produced lower cost values along with various amounts of sequences, depending on the configuration. The difference between these tests is the constrains on the first two planet variables showing how the limits places on variables impacts the results. The limits on continuous variables have

Table 5.4: Results from Different Archipelagos with Expanded Design Variable Limits

Model	Migrations	Min ($\frac{km}{s}$)	Avg ($\frac{km}{s}$)	Std	Num Seq
6 Islands V1	2	29.4	37.1	3.25	5
	4	22.1	37.0	4.21	6
	8	21.4	36.6	3.78	6
6 Islands V2	2	19.3	35.9	4.24	6
	4	19.6	36.8	4.27	5
	8	24.5	36.6	3.53	5
3 Islands	2	29.1	39.5	3.60	4
	4	25.8	39.1	3.97	6
	8	15.4	38.4	4.03	5

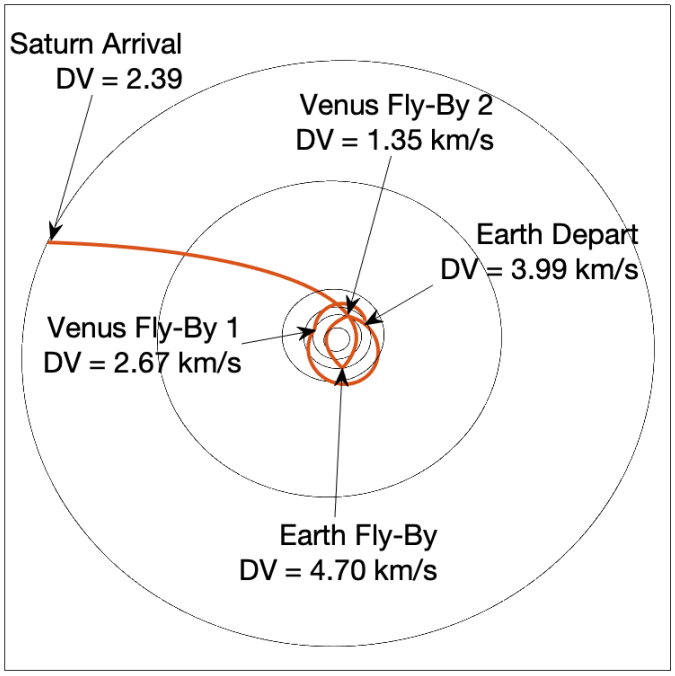

Figure 5.9: EVVES Trajectory Solutions from Best Island Model Result with Open Planet Constraints

a larger effect because in these tests the discrete limits were shortened which should have helped the solver find the more optimal solution space around 5 km/s. When the limits on the continuous variables are too wide, it decreases the performance of the solver because there are too many sequences to consider. This limitation is not overcome with the addition of collaboration through the island model for the extent

of these tests. The HGGA does exhibit improved results from the model parameters of migration and topology when parameters are constrained to an extent.

Chapter 6

CONCLUSION

This work demonstrated the effectiveness of the island model for improving the performance of different methods for solving MGA spacecraft trajectory optimization problems with undefined trajectory sequences. The HOCP was implemented to solve Branin's function and a simplified Mariner 10 mission. This method was very good at finding the best set of discrete variables for a given set of continuous ones because each set is solved with an independent algorithm. The collaboration of algorithms results in a long run time which is not ideal for scalability in the island model – a key factor. The HGGA allows for the combination of continuous and discrete variables within the same solver, making it a better fit for the island model. The HGGA results for Branin's function and the simplified Mariner 10 were not as good overall compared to HOCP, but some configurations produced solutions close to the know global optima.

The HGGA was implemented into different Archipelagos (island configuration) with the base algorithm settings derived from early testing to evaluate the GTOP Cassini 1 mission. The GTOP limits were altered to allow for four open fly-bys with tighter transfer time limits, and with two open fly-bys having the original GTOP transfer time limits. Comparing tests with the different limits revealed the effect design variable bounds have on the results in a the VSDS. When the planet bounds are too wide the solver is unable to converge on an optimal solution, even with the collaboration of the island model and tight time variable bounds. When the tighter planet bounds are used the solver produces more optimal cost values even with open time bounds. The bounds on the planets (discrete) have a larger effect on the results

than the transfer time (continuous) variables. In general, using this method for early phase mission design requires some knowledge of the desired solution with a degree of uncertainty around the encounter sequence. The natural selection and threshold selection methods explore different ranges of the solution space and can provide an improved solution when combined. An ideal number of migrations and larger number of islands may provides a more optimal cost along with a concentrated numbers of trajectory sequences. Other configurations result in an increased number of trajectory sequences with a less optimal answers. Depending on how much a given mission wants to consider other sequences, either result may be beneficial. Increasing the number of islands is more effective than changing the number of migrations at improving the solution but this also increases run time with the larger number of function evaluations. Changing the number of migrations is a more efficient way to improve solutions but requires preliminary testing. Before the island model parameters can be investigated the design variable limits must be set with knowledge of the scope of the problem at hand.

6.1 Future Work

The island model in this work had one type of base algorithm because of the focus on the HOCP and hidden gene functionality. Englander implemented other EAs as the HOCP base algorithm and the island model system has been thoroughly evaluated with EAs and other algorithms. Collaboration of different base algorithms is these cases improves the solution for a variety of problems with a defined trajectory sequence. Other EAs that have a similar population member structure may work with the hidden gene mechanism and it has already been implemented to a DE and PSO. Implementing hidden genes to more algorithms like EAs and ACOs that allows for the creation of heterogeneous Archipelagos that evaluate a VSDS. The island model

migration can be changed from synchronous to asynchronous, which would improve the run time of the model allowing for larger Archipelagos. MGADSM problems may also be evaluated with heterogeneous and homogeneous island models using resources from GTOP.

While there is some future work that may improve the capabilities of the hidden gene mechanism in an island model, it should be notes that the main influence on parameter tuning is the trajectory problem at hand. There is extensive research into the performance of island models that can be used for reference when tuning as the hidden gene mechanism does not change the nature of the algorithms in the island model. It would be most beneficial to work on this algorithm with actual problems as tuning with test problems may not translate to every problem. Working with this algorithm requires tuning but it can provide an efficient and reasonable solution to complex optimization problems.

BIBLIOGRAPHY

[1] Ackley Function.

[2] Statistical Software, Oct. 2018.

[3] Interplanetary exploration, Feb. 2019.

[4] Global Optimization Toolbox, Jan. 2020.

[5] O. Abdelkhalik. Autonomous Planning of Multigravity-Assist Trajectories with Deep Space Maneuvers Using a Differential Evolution Approach. *International Journal of Aerospace Engineering*, 2013:1–11, 2013.

[6] O. Abdelkhalik. Hidden Genes Genetic Optimization for Variable-Size Design Space Problems. *Journal of Optimization Theory and Applications*, 156(2):450–468, Feb. 2013.

[7] F. Biscani and D. Izzo. A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53):2338, Sept. 2020.

[8] F. Biscani, D. Izzo, W. Jakob, GiacomoAcciarini, M. Märtens, Michiboo, A. Mereta, C. Kaldemeyer, S. Lyskov, S. Corlay, M. V. Looz, B. Pritchard, A. Patel, M. López-Ibáñez, J. Mabille, G. Acciarini, K. Manani, A. Huebl, O. Webb, Jakirkham, Hulucc, J. Lee, A. Mambrini, Doodle1106, E. O'Leary, F. Lema, H. Nguyen, I. Smirnov, J. Jordan, and J. Travers. esa/pagmo2: pagmo 2.16.0, Sept. 2020.

[9] F. Biscani, D. Izzo, and C. H. Yam. A Global Optimisation Toolbox for Massively Parallel Engineering Optimisation. *arXiv:1004.3824 [cs, math]*, Apr. 2010. arXiv: 1004.3824.

[10] J. M. Bryan. GLOBAL OPTIMIZATION OF MGA-DSM PROBLEMS USING THE INTERPLANETARY GRAVITY ASSIST TRAJECTORY OPTIMIZER (IGATO). San Luis Obispo, California, Dec. 2011. California Polytechnic State University.

[11] M. Ceriotti and M. Vasile. Automated Multigravity Assist Trajectory Planning with a Modified Ant Colony Algorithm. *Journal of Aerospace Computing, Information, and Communication*, 7(9):261–293, Sept. 2010.

[12] M. Ceriotti and M. Vasile. MGA trajectory planning with an ACO-inspired algorithm. *Acta Astronautica*, 67(9-10):1202–1217, Nov. 2010.

[13] M. Clerc. *Particle swarm optimization*. ISTE, London ; Newport Beach, 2006. OCLC: ocm62782104.

[14] H. D. Curtis. Interplanetary Trajectories. In *Orbital Mechanics for Engineering Students*, pages 405–457. Elsevier, 2014.

[15] S. Dan and S. Robert H. Mariner 9, May 2020.

[16] S. A. Darani and O. Abdelkhalik. Space Trajectory Optimization Using Hidden Genes Genetic Algorithms. *Journal of Spacecraft and Rockets*, 55(3):764–774, May 2018.

[17] W. David R. Spacecraft Query, May 2020.

[18] W. David R. Venera 1, May 2020.

[19] K. Deb, N. Padhye, and G. Neema. Interplanetary Trajectory Optimization with Swing-Bys Using Evolutionary Multi-objective Optimization. In L. Kang, Y. Liu, and S. Zeng, editors, *Advances in Computation and Intelligence*, volume 4683, pages 26–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. Series Title: Lecture Notes in Computer Science.

[20] J. Devore, N. Farnum, and J. Doi. *Applied Statisticts for Engineers and Scientists*. Cengage Learning, 3rd edition edition, Jan. 2014.

[21] M. Dorigo and T. Stützle. *Ant colony optimization*. MIT Press, Cambridge, Mass, 2004.

[22] J. A. Dunne and E. Burgess. *The Voyage of Mariner 10 Mission to Venus and Mercury*. National Aeronautics and Space Administration Scientific and Technical Information Office Washington, D.C., Jet Propulsion Laboratory California Institute of Technology, Aug. 2004.

[23] J. Englander. Automated Trajectory Planning for Multiple-Flyby Interplanetary Missions. page 153.

[24] J. A. Englander, B. A. Conway, and T. Williams. Automated Mission Planning via Evolutionary Algorithms. *Journal of Guidance, Control, and Dynamics*, 35(6):1878–1887, Nov. 2012.

[25] J. A. Englander, M. A. Vavrina, and A. R. Ghosh. Multi-Objective Hybrid Optimal Control for Multiple-Flyby Low-Thrust Optimization. page 20, Williamsburg, VA, Jan. 2015. National Aeronautics and Space Administration Scientific and Technical Information Office Washington, D.C.

[26] T. J. Fitzgerald. Spacecraft Trajectory Optimization Suite (STOpS): Optimization of Multiple Gravity Assist Spacecraft Trajectories Using

Modern Optimization Techniques. San Luis Obispo, California, Dec. 2015. California Polytechnic State University.

[27] S. Fritz and K. Turkoglu. Optimal Trajectory Determination and Mission Design for Asteroid/Deep-Space Exploration via Multibody Gravity Assist Maneuvers. *International Journal of Aerospace Engineering*, 2017:1–12, 2017.

[28] D. Gaylor. GMAT.

[29] R. L. Haupt and S. E. Haupt. *Practical genetic algorithms*. John Wiley, Hoboken, N.J, 2nd ed edition, 2004.

[30] S. Hughes. GMAT Wiki - General Mission Analysis Tool (GMAT), Feb. 2015.

[31] S. Hughes, D. Conway, and J. Parker. Using the General Mission Analysis Tool (GMAT). 2017. Publisher: Unpublished.

[32] D. Izzo. Global Optimization and Space Pruning for Spacecraft Trajectory Design. In B. Conway, editor, *Spacecraft Trajectory Optimization*, pages 178–201. Cambridge University Press, Cambridge, 2010.

[33] D. Izzo and F. Biscani. Welcome to PyGMO — PyGMO 1.1.7dev documentation, 2015.

[34] D. Izzo, M. Ruciński, and F. Biscani. The Generalized Island Model. In F. Fernández de Vega, J. I. Hidalgo Pérez, and J. Lanchares, editors, *Parallel Architectures and Bioinspired Algorithms*, volume 415, pages 151–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. Series Title: Studies in Computational Intelligence.

[35] M. Mitchell. *An Introduction to Genetic Algorithms*. 1 edition, Feb. 1996.

[36] M. Märtens and D. Izzo. The Asynchronous Island Model and NSGA-II: Study of a New Migration Operator and its Performance. page 8.

[37] J. T. Olympio. DESIGNING OPTIMAL MULTI-GRAVITY-ASSIST TRAJECTORIES WITH FREE NUMBER OF IMPULSES. page 15.

[38] L. A. Ricciardi, C. A. Maddock, and M. Vasile. Direct Solution of Multi-Objective Optimal Control Problems Applied to Spaceplane Mission Design. *Journal of Guidance, Control, and Dynamics*, 42(1):30–46, Jan. 2019.

[39] S. Surjanovic and D. Bingham. Branin Function, 2013.

[40] A. C. Team. Global Trajectory Optimisation Problems Database, Aug. 2005. Section: Informatics, Mission Analysis.

[41] A. C. Team. Cassini 1, May 2013. Section: Informatics, Mission Analysis.

[42] Y. Tian, R. Cheng, X. Zhang, and Y. Jin. PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization. *arXiv:1701.00879 [cs]*, Jan. 2017. arXiv: 1701.00879.

[43] M. Vasile, E. Minisci, and M. Locatelli. Analysis of Some Global Optimization Algorithms for Space Trajectory Design. *Journal of Spacecraft and Rockets*, 47(2):334–344, Mar. 2010.

[44] T. Vinko and D. Izzo. Global Optimisation Heuristics and Test Problems for Preliminary Spacecraft Trajectory Design. page 9.

[45] T. Vinko, D. Izzo, and C. Bombardelli. Benchmarking different global optimisation techniques for preliminary space trajectory design. page 10.

[46] L. Wang, S.-w. Xiong, J. Yang, and J.-s. Fan. An Improved Elitist Strategy Multi-Objective Evolutionary Algorithm. In *2006 International Conference on Machine Learning and Cybernetics*, pages 2315–2319, Dalian, China, 2006. IEEE.

APPENDICES

Appendix A

BRANIN'S FUNCTION TESTING

Branin's Function Definitions:

$$\begin{aligned} f(\mathbf{x}^*) &= a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1-t)\cos x_1 + s \\ \mathbf{x} &= [x_1, x_2] \\ x_1 \in [-5, 10] \quad &and \quad x_2 \in [0, 15] \\ a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, r = 6, s = 10, t &= \frac{1}{8\pi} \end{aligned} \tag{A.1}$$

Three global optimum values: $f(\mathbf{x}^*) = 0.397887$

Table A.1: The 10 Different Test Configurations for Branin's Function Testing

	Uniform Crossover	Random Crossover
Random Selection	Rand-U	Rand-R
Natural Selection	NS-U	NS-R
Threshold Selection	T-U	T-R
Rank Weighted Selection	Rank-U	Rank-R
Cost Weighted Selection	Cost-U	Cost-R

Table A.2: Uniform Crossover Results for Different Selection Methods from Branin's Function Evaluated with HOCP

	Rand	NS	Thresh	Rank	Cost
Min	0.513873	0.399595	0.398239	0.527482	0.411203
%Err	0.0884	32.5708	3.3467	0.8830	0.0320
Avg	0.417915	0.398026	0.397897	0.405093	0.423193
%Err	5.0337	0.0349	0.0024	1.8111	6.3600
Std	0.0333	0.0001	0.0000	0.0061	0.0265
Avg C	1.0107	0.4497	0.4650	0.5883	1.0303
Avg run time	138.4228	196.3422	214.8980	198.2228	226.8760

Table A.3: Uniform Crossover Results for Different Selection Methods from Branin's Function Evaluated with HOCP

	Rand	NS	Thresh	Rank	Cost
Min	0.401400	0.398014	0.397982	0.419291	0.421271
%Err	0.8830	0.0320	0.0239	5.3794	5.8769
Avg	0.413430	0.398169	0.397896	0.406983	0.418633
%Err	3.9063	0.0708	0.0023	2.2861	5.2141
Std	0.0149	0.0003	0.0000	0.0089	0.0215
Avg C	0.7673	0.4597	0.4453	0.5713	0.8083
Avg run time	215.2308	236.8948	252.2705	235.1390	263.8878

Table A.4: Uniform Crossover Results for Different Selection Methods from Branin's Function Evaluated with HGGA

	Rand	NS	Thresh	Rank	Cost
Minimum	0.513873	0.399595	0.398239	0.527482	0.411203
%Err	29.1506	0.4293	0.0884	32.5708	3.3467
Average	2.111281	0.448729	0.405874	1.941790	2.579365
%Err	430.6233	12.7781	2.0074	388.0255	548.2656
Std	1.3755	0.0716	0.0095	1.3697	1.8811
Avg C	0.1000	0.0000	0.0000	0.0667	0.2333
Avg run time	0.0582	0.0627	0.0651	0.0701	0.0781

Table A.5: Random Crossover Results for Different Selection Methods from Branin's Function Evaluated with HGGA

	Rand	NS	Thresh	Rank	Cost
Minimum	0.401400	0.398014	0.397982	0.419291	0.421271
%Err	0.8830	0.0320	0.0239	5.3794	5.8769
Average	1.535585	0.428899	0.403044	1.496898	2.271577
%Err	285.9350	7.7942	1.2961	276.2119	470.9101
Std	1.3037	0.0319	0.0047	1.1987	1.7363
Avg C	0.0333	0.0000	0.0000	0.0333	0.1333
Avg run time	0.0779	0.0722	0.0839	0.0813	0.0895

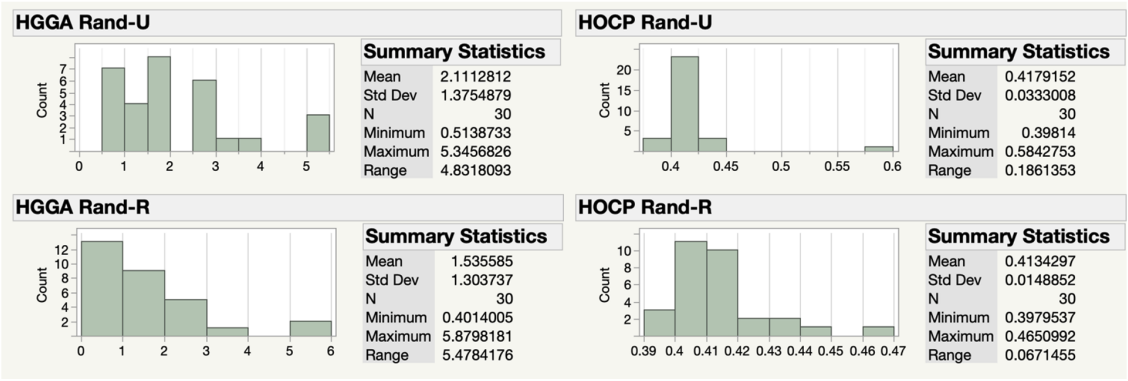

Figure A.1: Random Selection Distributions for HGGA and HOCP Evaluating Branin's Function

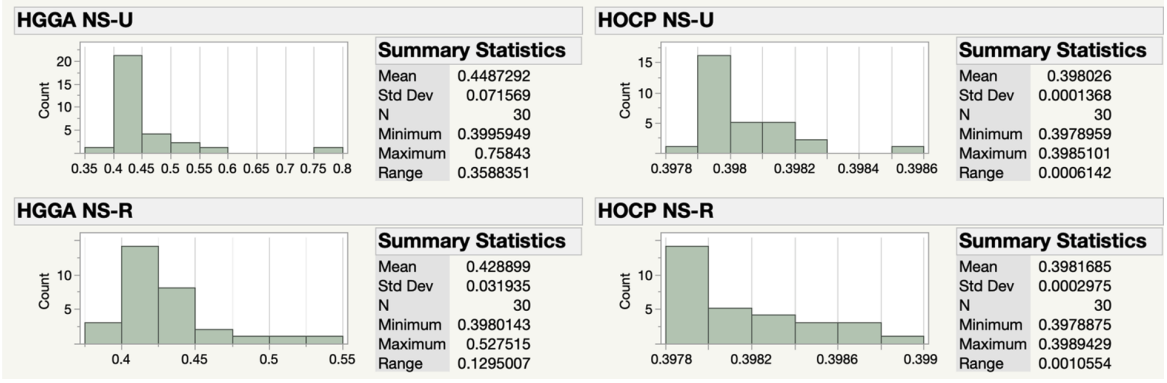

Figure A.2: Natural Selection Distributions for HGGA and HOCP Evaluating Branin's Function

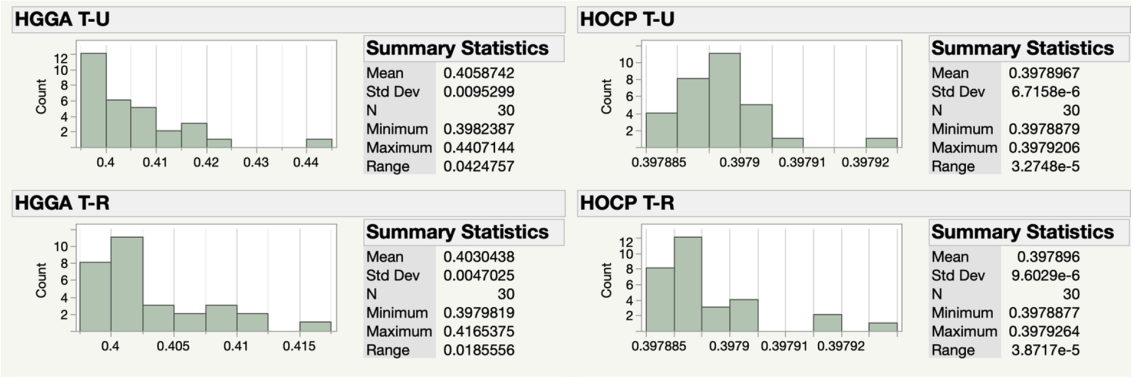

Figure A.3: Threshold Selection Distributions for HGGA and HOCP Evaluating Branin's Function

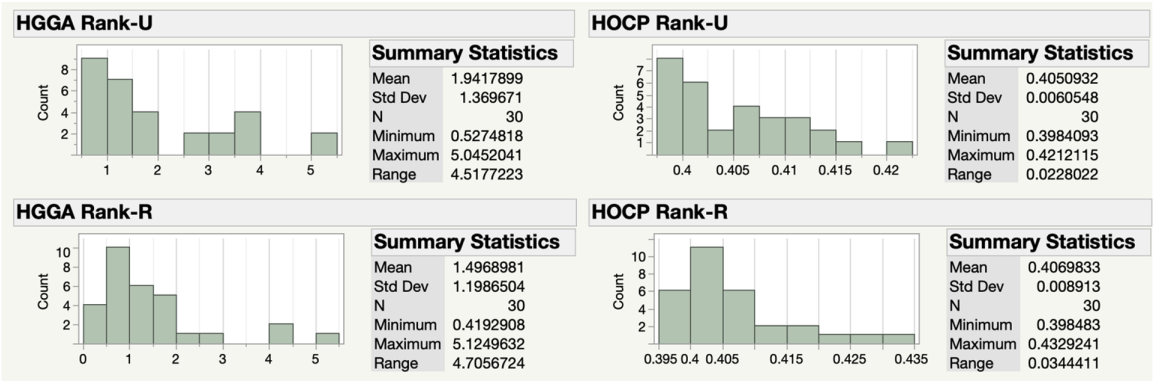

Figure A.4: Rank Weighted Selection Distributions for HGGA and HOCP Evaluating Branin's Function

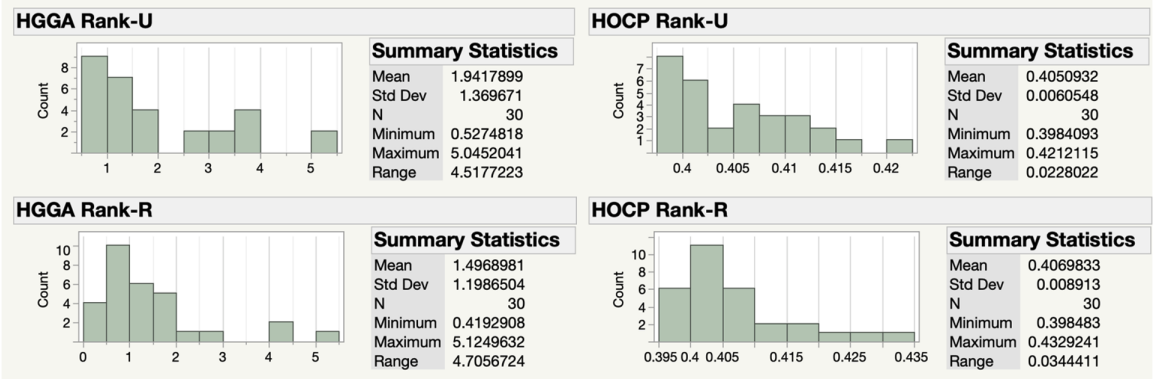

Figure A.5: Cost Weighted Selection Distributions for HGGA and HOCP Evaluating Branin's Function

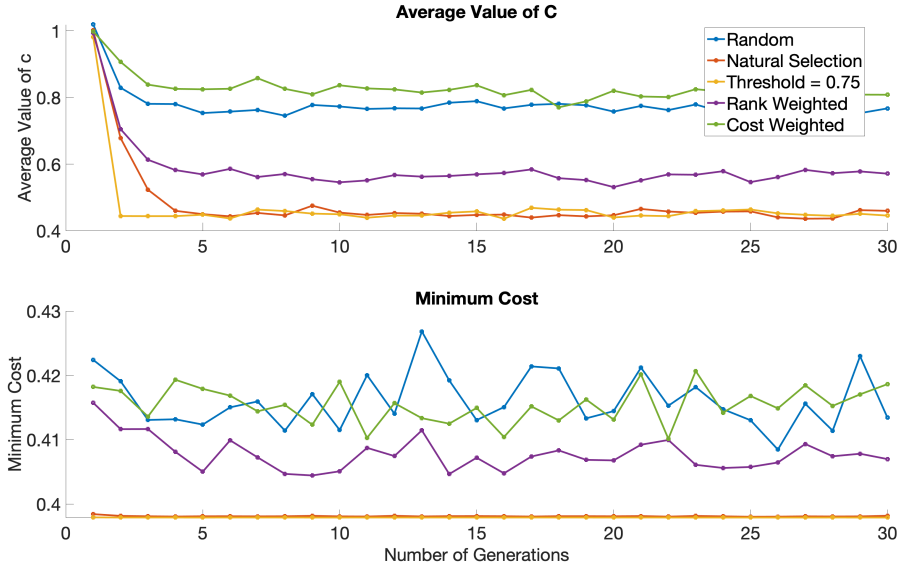

Figure A.6: Trend of Branin's Function with Random Crossover HOCP

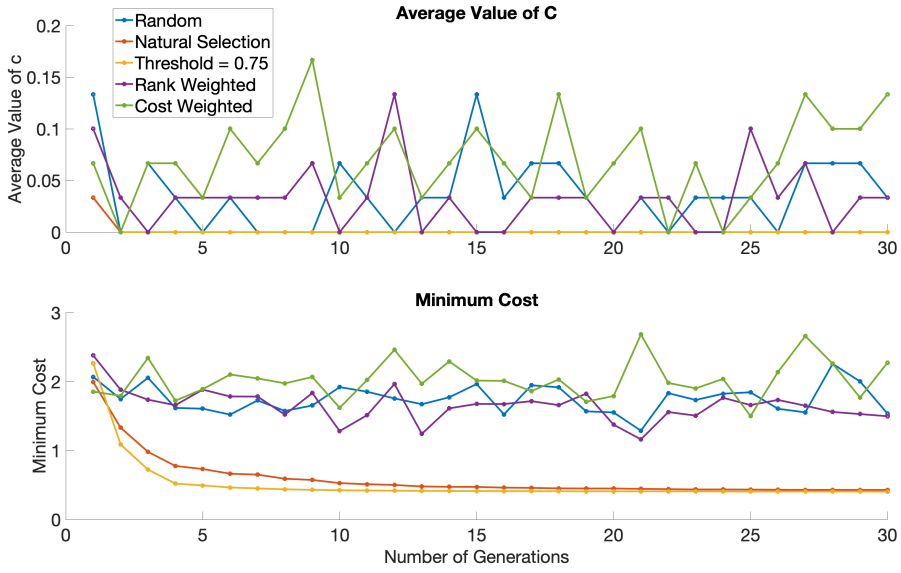

Figure A.7: Trend of Branin's Function with Random Crossover HGGA

Appendix B

MARINER 10 TESTING

[Mariner 10 Mission]Mariner 10 Mission

	Reported Mariner 10 [22]	Simple Mariner 10 [26]
Fly-By Planet 1	V	V
Fly-By Planet 2	M	-
Fly-By Planet 3	M	-
Fly-By Planet 4	M	-
Departure Date (UTC)	1973-11-03	1973-11-03
T_1 (days)	94	94
T_2 (days)	52	52
T_3 (days)	237	-
T_4 (days)	115	-

Table B.1: HOCP ΔV Results for Different Selection and Crossover Methods

HOCP Arrival Delta-V Included

	Avg (km/s)	Std	Min (km/s)	Dep Date UTC	t_1	t_2	t_3
NSU	11.2571	0.0754	11.1566	28-Nov-1973	101	65	67
NSR	8.3796	4.5661	1.6961	29-Nov-1973	81	86	65
TU	9.0278	3.2343	1.5671	27-Nov-1973	107	62	66
TR	11.1778	0.0489	11.1263	25-Nov-1973	99	69	68

HOCP No Arrival Delta-V

	Avg (km/s)	Std	Min (km/s)	Dep Date UTC	t_1	t_2	t_3
NSU	1.8253	0.1301	1.6433	24-Nov-1973	105	67	66
NSR	4.5478	4.5539	1.5176	11-Nov-1973	92	94	66
TU	5.3529	4.9221	1.2868	30-Nov-1973	104	63	64
TR	1.9346	0.08623	1.7549	28-Nov-1973	104	64	66

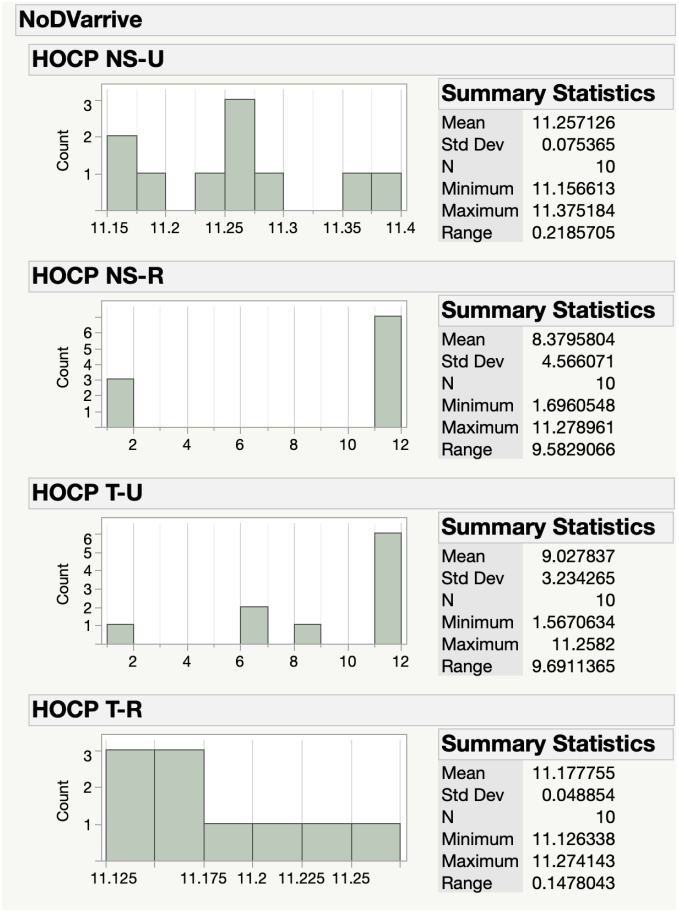

Figure B.1: Distributions from Mariner 10 Samples with the Arrival Delta-V Excluded from the Solutions

Table B.2: HOCP ΔV Component Results for Different Selection and Crossover Methods

HOCP Arrival Delta-V Included

	ΔV_{depart}	ΔV_{fb1}	ΔV_{fb2}	ΔV_{arrive}
NSU	9.528797	0.002026	0.057565	1.701026
NSR	9.678286	0.004516	0.008662	1.504460
TU	9.834196	0.010564	0.026535	1.249736
TR	9.632275	0.001913	0.051491	1.473301

HOCP No Arrival Delta-V

	ΔV_{depart}	ΔV_{fb1}	ΔV_{fb2}	ΔV_{arrive}
NSU	0.014519	9.470721	0.022489	1.648884
NSR	0.021639	0.004654	0.000948	1.668814
TU	0.008586	0.050379	0.005079	1.503019
TR	0.006308	9.461727	0.004328	1.653976

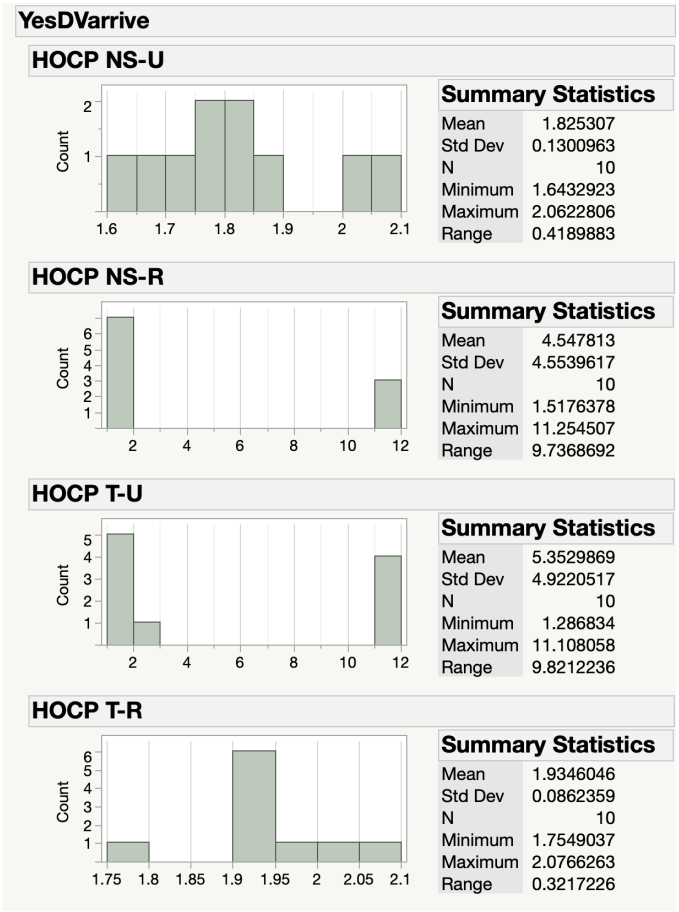

Figure B.2: Distributions from Mariner 10 Samples with the Arrival Delta-V Included in the Solutions

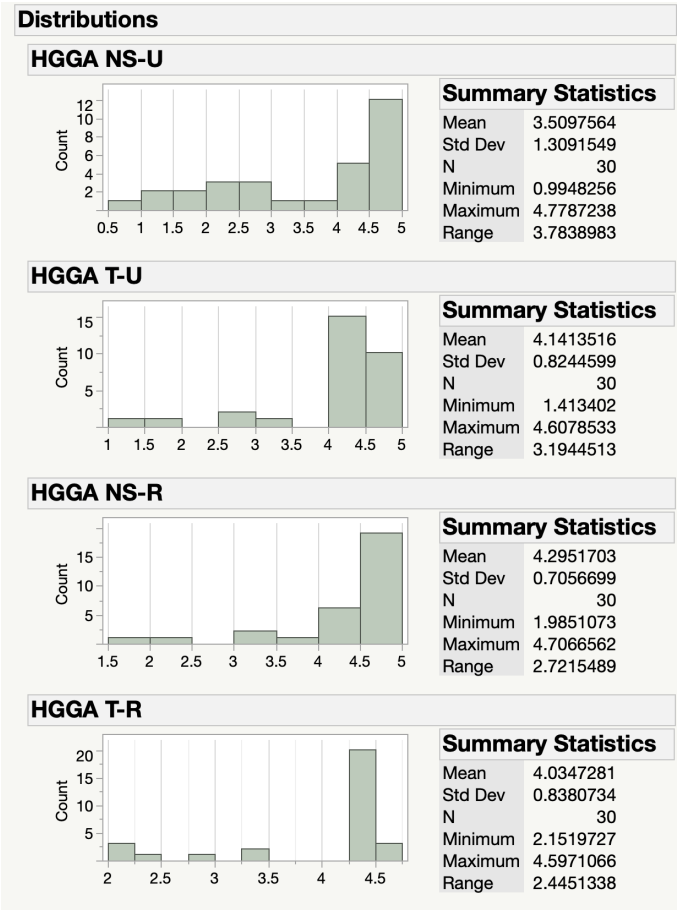

Figure B.3: Distributions of Samples for HGGA Testing

Table B.3: HGGA ΔV Results for Different Selection and Crossover Methods

HGGA - All Samples

	Avg (km/s)	Std	Dep Date UTC	t_1	t_2	t_3
NSU	3.5098	1.3092	1973-11-30	104	68	58
NSR	4.1414	0.8245	1973-11-23	107	68	64
TU	4.2952	0.7057	1973-11-11	110	70	70
TR	4.0347	0.8381	1973-11-29	97	63	71

HGGA - EVM

	Avg (km/s)	Std	Dep Date UTC	t_1	t_2	t_3
NSU	4.5414	0.0926	1973-11-03	90	58	-
NSR	4.4825	0.0502	1973-11-06	87	56	-
TU	4.5794	0.0883	1973-11-01	92	57	-
TR	4.4694	0.0514	1973-11-04	89	57	-

HGGA - EEEM

	Avg (km/s)	Std	Dep Date UTC	t_1	t_2	t_3
NSU	3.5098	1.3092	1973-11-30	104	68	58
NSR	4.1414	0.8245	1973-11-23	107	68	64
TU	4.2952	0.7057	1973-11-11	110	70	70
TR	4.0347	0.8381	1973-11-29	97	63	71

Table B.4: HGGA ΔV Component Results for Different Selection and Crossover Methods

HGGA - All Samples

	Min (km/s)	ΔV_{depart}	ΔV_{fb1}	ΔV_{fb2}	ΔV_{arrive}
NSU	0.9948	0.017452	0.001682	0.975690	10.647380
NSR	1.4134	0.008662449	0.010795	1.393943	9.825043
TU	1.9851	0.017424353	0.011204	1.956478	9.448132
TR	2.1519	0.005081768	0.047235	2.099655	9.171176

HGGA - EVM

	Min (km/s)	ΔV_{depart}	ΔV_{fb1}	ΔV_{fb2}	ΔV_{arrive}
NSU	4.3741	4.3467	0.027395	-	10.240082
NSR	4.3997	4.3839	0.015768	-	10.146120
TU	4.4043	4.4041	0.000183	-	10.172403
TR	4.3988	4.3790	0.019864	-	10.172074

HGGA - EEEM

	Min (km/s)	ΔV_{depart}	ΔV_{fb1}	ΔV_{fb2}	ΔV_{arrive}
NSU	0.9948	0.017452	0.001682	0.975690	10.647380
NSR	1.4134	0.008662449	0.010795	1.393943	9.825043
TU	1.9851	0.017424353	0.011204	1.956478	9.448132
TR	2.1519	0.005081768	0.047235	2.099655	9.171176

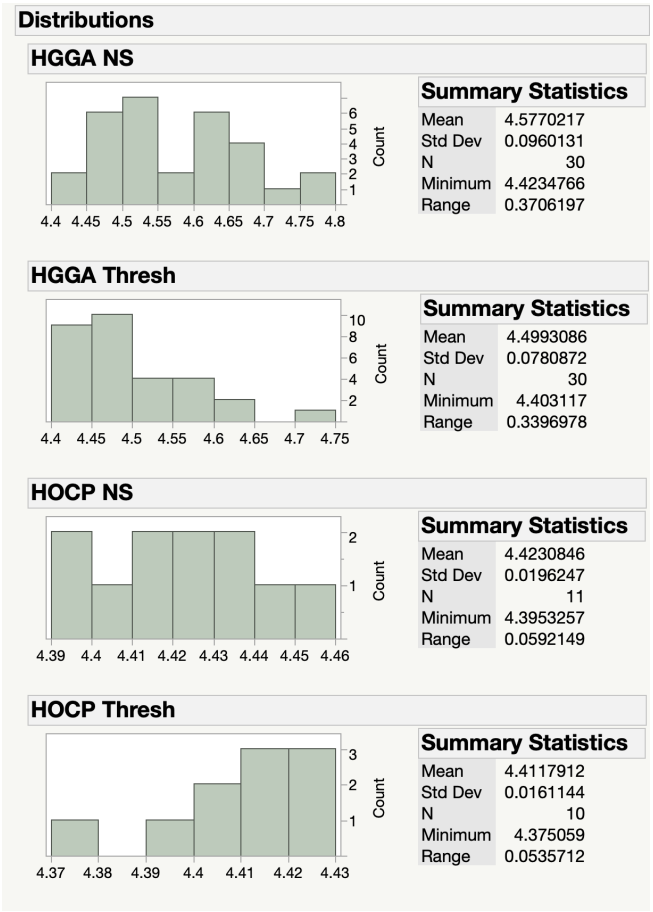

Figure B.4: Constrained Mariner 10 Distributions for Both HGGA and HOCP

Appendix C

CASSINI 1

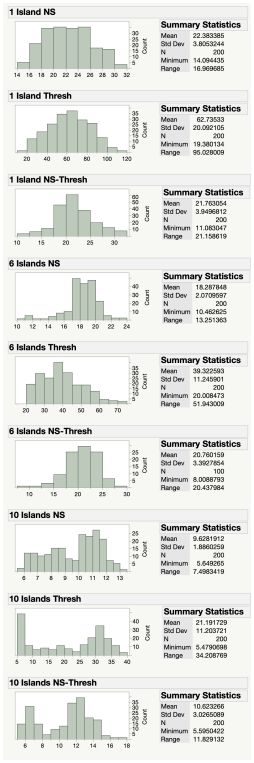

Figure C.1: Distributions of Solutions Sets with Number of Islands Varied

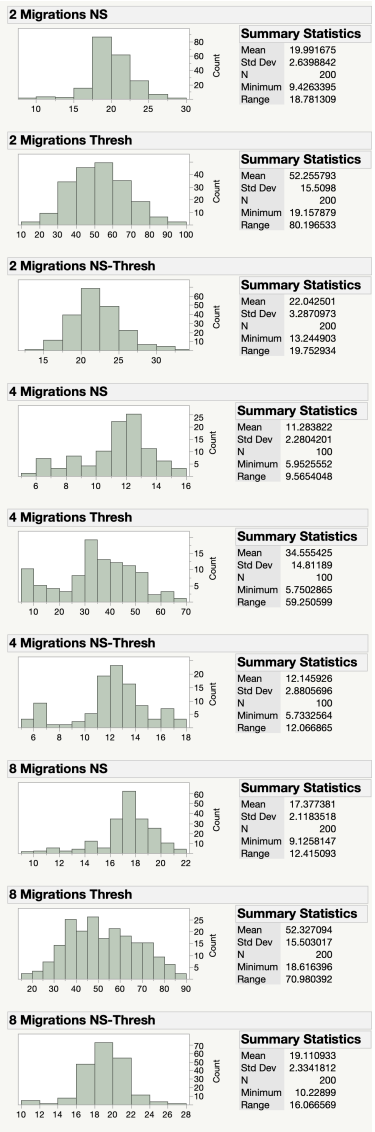

Figure C.2: Distributions of Solutions Sets with Number of Migrations Varied

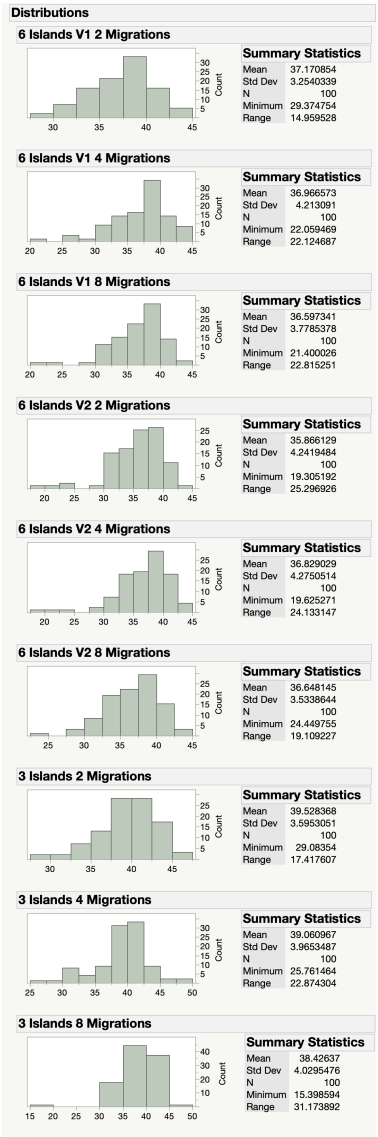

Figure C.3: Distributions of Solutions Sets with Planet Design Variable Limits Expanded