A HYBRID GENETIC ALGORITHM APPROACH TO GLOBAL LOW-THRUST TRAJECTORY OPTIMIZATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Matthew A. Vavrina

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2008

Purdue University

West Lafayette, Indiana

For Mom, Dad, and Patty

ACKNOWLEDGMENTS

First, and foremost, I want to thank my advisor, Professor Kathleen Howell, for her mentorship and guidance. Her inexhaustible passion is contagious and indelible; I will take it with me wherever I go. I am also grateful to Professor James Longuski for his sage advice and open door. Many thanks are also due Professor William Crossley for his inspiring lectures and introducing me to genetic algorithms. It was during his class that the fundamental idea for this research was conceived.

At Purdue, I have been fortunate to be part of a research group composed of an amazing assemblage of individuals. I am thankful for the friendship of Todd Brown, Diane Craig Davis, Dawn Gordon, Dan Grebow, Lucia Irrgang, Masaki Kakoi, Dr. Lindsay Millard, Zubin Olikara, Marty Ozimek, Chris Patterson, Tom Pavlak, Raoul Rausch, Wayne Schlei, Cody Short, Mar Vaquero, and last, but not least, Geoff Wawrzyniak.

I am also extremely appreciative of Dr. Jon Sims for the opportunity to work at JPL and hooking me on low-thrust trajectory design. I thank Dr. Chit Hong Yam for providing invaluable guidance on the GALLOP software incorporated in this investigation. Additionally, I am thankful to the School of Aeronautics and Astronautics at Purdue University for financial support.

I am forever grateful to my parents and family; their love and support is limitless. Most importantly, I thank my wife, Patty. It is a joy to share this life with you.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Vavrina, Matthew A. M.S.A.A., Purdue University, December, 2008. A Hybrid Genetic Algorithm Approach to Global Low-Thrust Trajectory Optimization. Major Professor: Kathleen C. Howell.

To expand mission capabilities that are required for exploration of the solar system, methodologies to design optimal low-thrust trajectories must be developed. However, low-thrust, multiple gravity-assist trajectories pose significant optimization challenges because of their expansive, multimodal design space. In this work, a technique is developed for global, low-thrust, interplanetary trajectory optimization through the hybridization of a genetic algorithm and a gradient-based direct method (GALLOP). The hybrid algorithm combines the effective global search capabilities of a genetic algorithm with the robust convergence and constraint handling of the local, calculus-based direct method. The automated approach alleviates the difficulty and biases of initial guess generation and provides near globally-optimal solutions. Both single objective and multiobjective implementations are developed. In the single objective implementation, the technique is applied to several complex low-thrust, gravity-assist trajectory scenarios, generating previously unpublished optimums. Specifically, the single objective hybrid algorithm generates apparent global optimums for a direct trajectory scenario to Mars, as well as gravity-assist trajectories with three intermediate flybys to Neptune and Pluto. The multiobjective implementation incorporates the NSGA-II algorithm to generate a Pareto front of solutions that are globally optimal in terms of both final delivered mass and time-of-flight in a single execution. The multiobjective hybrid algorithm is applied to a direct Earth-Mars rendezvous design scenario, successfully developing a Pareto front of near-globally optimal trajectories, enabling a tradeoff decision on the two objectives.

1. INTRODUCTION

Curiosity has driven mankind to explore beyond the safe confines of Earth's atmosphere. Probes have been dispatched throughout the solar system to better understand the universe, its laws, and possibly discover extraterrestrial life. These robotic explorers, while dramatically uncovering many mysteries, have exposed merely a fraction of the Solar System and mostly spurred new questions. To answer these questions future missions will require larger scientific payloads, more flexibility, and likely take humans to other planets. Much of this burden falls upon the mission designer. Trajectories that meet increasingly demanding mission objectives must be developed to enable further discoveries.

Through developments in electric propulsion, low-thrust trajectories have emerged as efficient alternatives to chemically-propelled trajectories for interplanetary missions. The high specific impulse, I_{sp}, associated with low-thrust, electric propulsion systems allows higher exhaust velocities as compared to conventional chemical propulsion; thus, larger changes in spacecraft velocity are accomplished with the same amount of propellant. This improved efficiency can reduce the required propellant, increasing deliverable mass. Additionally, low-thrust trajectories are generally less sensitive to missed thrusting and possess a broader range of launch opportunities. Electric propulsion technologies have been successfully demonstrated on several recent interplanetary missions, including the National Aeronautics and Space Administration (NASA) missions Deep Space 1 [1] and Dawn [2], as well as the Japan Aerospace Exploration Agency's Hayabusa [3] mission.

Additional reductions in required propellant for the challenge of interplanetary flight can be attained by implementing gravity assists from close planetary flybys. During a close encounter, the gravity of both the spacecraft and planet affects the trajectory of the other body; however, the path of the spacecraft is altered much more significantly

because of its relatively small mass. If the encounter geometry is designed correctly, the planet imparts a substantial, and effectively "free", change in velocity of the spacecraft. Historically gravity assists have been incorporated into a number of chemically propelled trajectories (e.g., Mariner 10, Voyager 1, Galileo, and Cassini), often enabling flight scenarios by significantly increasing deliverable mass and decreasing flight time. The previous trajectory designs were all based on the availability of chemical propulsion for limited maneuver capability. Lawden [4], Crocco [5], and Breakwell [6], among others, conducted pioneering work in advancing the gravity-assist concept for interplanetary spacecraft in the mid-1950's and early 1960's.

The combination of high-I_{sp}, low-thrust propulsion and gravity-assist maneuvers often improves the delivered mass, time-of-flight (TOF), and flexibility of interplanetary missions [7]. However, the potential for superior performance is accompanied by a significant increase in optimization complexity. Low-thrust trajectories typically involve long periods of continuous thrusting, adding significant complications to the problem formulation and solution strategy. The addition of multiple gravity-assist bodies, while offering the potential for significant propellant reduction, further exacerbates optimization complexity because of phasing constraints [8]. The challenges of low-thrust, gravity-assist (LTGA) trajectory optimization are substantial, and a variety of local optimization techniques has been explored. To fully exploit the benefits of electric propulsion and gravity assists, however, effective *global* optimization techniques are required.

1.1 Low-Thrust Trajectory Optimization

Trajectory optimization is the original impetus behind the field of optimal control. Johann Bernoulli's brachistochrone problem (1696) prompted the development of the calculus of variations, which was refined by Leonhard Euler and Joseph Lagrange in the 1700's. The introduction of high-speed computing in the 20th century triggered an upsurge of research into finite-burn trajectory optimization for spacecraft. The publication of Lawden's primer vector theory [9] in 1963 was fundamental in expanding

the development of techniques to generate locally optimal trajectories. As computing speeds increased, higher dimensional problems could be solved numerically, and new approaches exploiting nonlinear programming techniques emerged. As the investigations into new strategies have broadened in scope, an ever-widening range of techniques is bringing new capabilities and perspective to the problem. Thus, the current state-of-the-art in low-thrust trajectory optimization encompasses a wide array of methods that are typically categorized as either indirect or direct [10].

Indirect methods involve formulations from the calculus of variations, and yield an exact solution to the low-thrust optimization problem with only a few design variables [11,12]. The Euler-Lagrange theorem, along with Pontryagin's Minimum Principle, are applied to create a two-point boundary value problem (TPBVP) that must be solved to obtain the continuous optimal thrust history. (See Bryson and Ho [13].) Two distinct advantages of indirect methods are the low dimensionality of the design space and the high fidelity of the solution. However, the formulation requires an initial guess for physically nonintuitive costates, or Lagrange multipliers. Additionally, indirect strategies are exceptionally sensitive to the initial guess, creating a highly multimodal design space [14]. These sensitivities are amplified significantly if flyby bodies are incorporated into the trajectory.

In contrast to indirect approaches, direct methods parameterize the continuous thrust, incorporating approximations into the trajectory model. This formulation results in a parameter optimization problem, and nonlinear programming techniques can be applied to solve the problem. Unlike indirect methods, in direct approaches the only design variables required are physically intuitive. However, the parameterization of the thrust often generates many design variables, creating an expansive design space of high dimensionality. Despite the significant increase in variables and a limited accuracy, direct techniques are much more robust in response to an initial guess than indirect methods. This increased radius of convergence renders direct strategies capable of optimizing low-thrust trajectories with several gravity assists. Nonetheless, an initial guess that is sufficiently close to a local optimal is required for convergence, and convergence properties vary depending on the formulation. Direct formulations

implemented for low-thrust trajectory optimization include: differential inclusion [15,16], collocation [17,18], direct transcription [19], Whiffen's Static/Dynamic Control algorithm [20], and various other schemes [21]. Some techniques do not fit perfectly into either a direct or an indirect categorization. Such is the case with techniques that combine both methods [22, 23].

Two key drawbacks inherent to both indirect and direct methods (in traditional calculus-based formulations) are: (1) the lack of a global search capability; and (2) the requirement of a suitable initial guess.

1.2 Previous Approaches to Global Low-Thrust Trajectory Optimization

Global optimization of low-thrust, gravity-assist trajectories is a difficult task. The only method that can guarantee the generation of a global optimum for a problem with such a complex design space is an exhaustive grid search over each design variable at a resolution that is sufficiently small. Such a search is impossible to complete within a reasonable time period for even modestly sized LTGA problems with current computing speeds (even if parallelized). The impracticality of a grid search is due to the high dimensionality in a direct approach and the extreme sensitivity of the solution to the costates in an indirect formulation. The expansive design space is also highly multimodal, with many local optima permeating a mostly infeasible fitness landscape. Furthermore, the complexity and size of the design space increases significantly with the addition of intermediate bodies to the flyby sequence.

Several recent approaches to low-thrust, gravity-assist trajectory optimization incorporate global search schemes. Petropoulos and Longuski developed an effective and efficient search technique with the software STOUR-LTGA [24,25]. In STOUR-LTGA, tangential thrusting is assumed, and low-thrust trajectories are represented kinematically in terms of an exponential sinusoid shape, allowing an analytical expression for the thrust profile. Without the requirement of integrating the trajectory, an efficient, yet broad, grid-like survey of the design space can be accomplished. This procedure in STOUR-LTGA often generates thousands of feasible trajectories. High performing candidate trajectories

are then selected and applied as an initial guess for local optimization in a direct method. The effectiveness of this two-step approach has prompted other researchers to develop other shape-based strategies [26-28]. While successful in generating a high-performing initial guess, the assumption of a specific shape, which is fundamental to these approaches, is limited to trajectories with tangential or radial thrust and a particular thrusting structure (e.g., thrust-coast or coast-thrust). Therefore, the optimization of the initial guess is biased towards those types of trajectories.

Several other recent global optimization techniques employ evolutionary computation [29] for low-thrust, interplanetary trajectories. Dachwald [30] and Carnelli et al. [31] have experimented with evolutionary neurocontrollers, while De Pascale and Vasile [27] have combined a shape-based strategy with an evolutionary algorithm. Other researchers have focused on the combination of genetic algorithms with indirect methods, including Hartmann et al. [32], Wuerl et al. [33], and Woo et al. [34], along with Sentinella and Casalino [35]. The development by Hartmann et al. incorporates a multiobjective genetic algorithm combined with indirect-based software, namely the Solar Electric Propulsion Trajectory Optimization Program (SEPTOP) [36]. Final mass and time-of-flight are simultaneously optimized for direct interplanetary trajectories to Mars and Mercury. Wuerl et al. followed this work with a single objective hybridization of an indirect method and a genetic algorithm to optimize a Mars sample return mission. Similarly, Woo et al. extend the combination of a genetic algorithm and SEPTOP to a single objective implementation for optimization of single gravity-assist, SEP trajectories to the outer planets. However, the hybrid algorithm is applied over a reduced design space, by characterizing previously known optimal solutions based on solar revolutions, propulsion efficiency, and a ballistic trajectory approximation. As a variation on the other implementations of an indirect-GA combination, Sentinella and Casalino incorporate a genetic algorithm to solve the TPBVP arising from the calculus of variations formulation.

1.3 Focus of this Work

The strategy in this development is a technique that determines near-globally optimal, low-thrust, gravity-assist trajectories through the hybridization of a genetic algorithm and a local gradient-based direct method. This approach exploits propellant-efficient low-thrust propulsion by effectively searching the entire design space without the difficulties and biases of initial guess generation. The coupling is structured so that the pitfalls of both the genetic algorithm and the direct method applied individually are avoided, and the advantageous properties of each are emphasized. The calculus-based direct method is integrated into the genetic algorithm, efficiently refining the designs in the population, while the genetic algorithm globally explores the design space. This search technique cannot guarantee the generation of the absolute global optimum, but the combination provides a synergy that leads to an effective global search.

This study is organized as follows:

- Chapter 2: The development of an effective hybrid algorithm, capable of globally optimizing low-thrust, gravity-assist trajectories, requires some background information. A genetic algorithm, the global optimization component of the hybrid, is introduced. Genetic algorithms are capable of either single objective or multiobjective optimization in one execution of the algorithm, and both implementations are detailed. The local component of the hybrid algorithm, based on the Purdue software Gravity-Assist Low-thrust Local Optimization Program (GALLOP), is also described. GALLOP incorporates a trajectory model and a calculus-based optimizer that, together, afford the software the capability to locally optimize low-thrust trajectories with many intermediate bodies efficiently.

- Chapter 3: The advantages of the hybrid formulation are described, motivating the combination a genetic algorithm and GALLOP. Several hybrid design considerations and implications are detailed and then the single objective and multiobjective implementations are described. Although the hybrid algorithm does not require an

initial guess, the genetic algorithm necessitates some user input. These necessary parameters and the design variables of algorithm are also outlined.

- Chapter 4: Results of application of both the single objective and multiobjective implementations of the hybrid algorithms are presented. The single objective implementation is applied to several complex interplanetary low-thrust trajectory examples, successfully generating previously unknown optimums. Specifically, a direct trajectory scenario with solar electric propulsion (SEP) to Mars is examined, as well as gravity-assist trajectories to Neptune and Pluto over a wide range of launch and arrival dates. The multiobjective implementation is applied to a SEP Earth-Mars rendezvous design scenario. An entire set of solutions is generated. The solutions are near-globally optimal in terms of both final mass and time-of-flight. The results from both implementations are compared with previously published results.

- Chapter 5: The study is summarized and recommendations for future research are outlined in this final chapter.

2. BACKGROUND

2.1 Genetic Algorithms

A genetic algorithm (GA) is a stochastic optimization method based on the biological principles of Darwinian evolution [37]. Genetic algorithms incorporate operators that mimic natural selection and reproduction (on a simplistic level) using a probabilistic search on a population of designs. The population 'evolves' through the application of genetic operators to determine the design that is best adapted to a fitness landscape. The fitness landscape is defined by a fitness function that is typically composed of an objective function and an appended penalty function if the problem is constrained. Unlike calculus-based methods, GAs are developed as a framework for a global search of the design space [37]. That is, in a GA, the potential paths are not confined to the locally optimal solution in the neighborhood of an initial guess; complex, expansive, multimodal design spaces can be explored. In addition to the global optimization capability, genetic algorithms do not necessitate a user-defined initial guess, and, thus, possess two fundamental characteristics that are required in the global component of the hybrid algorithm for low-thrust trajectory optimization.

2.1.1 Overview

Genetic algorithms were formulated by Holland in 1975 as a computational technique to artificially model biological evolution [38]. Genetic algorithms as search and optimization routines were popularized by Goldberg's 1989 publication *Genetic Algorithms in Search, Optimization, and Machine Learning* [39]. The GA and its variations, along with similarly inspired genetic programming, evolutionary

programming, and evolution strategies, constitute the family of evolutionary algorithms (EA) [40]. Evolutionary algorithms are then grouped into the larger category of biologically-motivated evolutionary computation, which includes techniques such as ant colony optimization, particle swarm optimization, and differential evolution.

As computational speeds have increased, the use of genetic algorithms has expanded to a wide variety of applications including numerical and combinatorial optimization problems in engineering. The technique is effective and efficient for difficult optimization problems where the solution space is nonconvex, multimodal, or discontinuous. The GA is a distinctly different approach to optimization when compared to traditional methods, and its unique formulation affords it successful implementation on a wide variety of problems. However, the GA is no panacea and is best suited for complex problems. It is computationally expensive, often requiring many function evaluations because of its population-based search. For continuous, unimodal problems or problems in which a local search is acceptable, traditional calculus-based techniques can be significantly more efficient. Calculus-based methods typically require fewer function evaluations because of their use of gradient information in a point-to-point search. Furthermore, because GAs do not use gradients, there is no proof of convergence such as the Kuhn-Tucker conditions for calculus-based methods [41]. The evaluation of convergence in GAs is more ambiguous, where different executions of the algorithm can produce different final solutions for the same problem setup. Additionally, GAs cannot always achieve a high degree of precision because of the encoding of the design variables to a finite resolution. Moreover, genetic algorithms were not originally designed for unconstrained problems, and without gradient information may not be effective on tightly constrained problems with small feasible regions.

Despite the drawbacks, GAs are powerful optimization techniques for problems that are too complex to employ traditional methods. Its population-based formulation, though computationally expensive, is key to its efficacy. The GA incorporates probabilistic transition rules on a population instead of deterministic rules on a single design (i.e., calculus-based approaches). Hence, GAs are capable of searching over “rugged” and discontinuous fitness landscapes. Instead of using derivatives to determine a search

direction, genetic algorithms use the fitness value of each individual in the population to identify areas of the design space that warrant further exploration. Without the requirement of gradient information, GAs can be applied to problems in which derivatives are not available, such as combinatorial or mixed problems. The population-based approach also affords GAs their ability to avoid 'getting stuck' in local optima. Working with many designs simultaneously, the GA search process can globally explore multimodal design spaces, searching over many local optima for the global optimum.

2.1.2 Binary Representation

In addition to a population-based search, another key difference between traditional optimization techniques and GAs is the use of encoded design variables. In a simple genetic algorithm (SGA) [39], each design variable is represented by a binary string comprised of 1's and 0's. The binary string is the SGA's the artificial analog of a *gene*. Genes are then concatenated to form a *chromosome*, sometimes designated the *genotype*, which represents the values of all the design variables of an individual in the population. To distinguish between the quality of each design in the population (i.e., how well adapted to the design environment) the chromosome must be decoded and the objective function evaluated for the determination of the fitness value of the individual. The decoded, real-valued chromosome is the analog of the *phenotype*, the observable characteristics of an organism, in the SGA.

As an example, consider a function, $f(x,y)$, to be optimized, where x and y are the design variables. If the binary string length for each variable is specified as 5 bits, the gene for x could be 01001 and y could be 11010, both representations of real-valued numbers, resulting in the chromosome: 0100111010. The binary format of the chromosome enables the genetic operators to combine and modify genes to discover and promote patterns in the chromosome that are best adapted to the fitness landscape. This binary representation is inherently discrete. Continuous variables are discretized to a certain resolution r_i:

$$r_i = \frac{x_i^U - x_i^L}{2^{b_i} - 1} \tag{2.1}$$

where x^U and x^L are the user-defined upper and lower bound of the i^{th} design variable and b_i is the number of bits used to code the design variable as a gene. Binary coding permits the evaluation of both discrete and continuous variables. However, a coarse-grained search on continuous variables can limit the accuracy the solution. The larger the range between the upper and lower bound for a design variable, the more bits required to achieve a desired precision; and, therefore, the more iterations the GA will require to converge. Thus, machine precision may require an unwieldy number of bits.

A binary-coded GA is used for this analysis, although different implementations also exist. Real-coded GAs, where each gene is a real-valued variable, have been investigated and applied with success [39]. However, Goldberg's underlying theory for the GA's mathematical foundation does not correspond to real-valued parameters in the genetic operators. An additional variation that is often implemented is the incorporation of a Gray coding, instead of a standard binary coding, for the design variables. Gray coding, or reflected binary coding, is another base 2 numbering system that maintains the distance between consecutive values to a single bit flip. Such a coding can aid in refining solutions more efficiently.

2.1.3 Population Initialization

One of the greatest advantages of genetic algorithms is that no user-defined initial guess is required to start the optimization process. For many problems, especially low-thrust, gravity-assist trajectory optimization, developing a suitable initial guess is an exceedingly difficult step in the optimization procedure. Genetic algorithms can be initiated without any prior knowledge of the design space or an auxiliary technique to develop an initial guess. The initial population can be generated entirely randomly by using a random number generator to define the bits of the chromosome. This

initialization step essentially represents a coin flip to determine whether each bit location in a chromosome will receive a 1 or a 0. It is possible to seed the population with specified designs to bias the search to predetermined locations of the design space. However, this is not necessary and is often not desirable. The random approach to define the initial population ensures a fair and broad initial sampling of the design space (if using a uniform random number generator). Therefore, the only data required to initiate the optimization process are the definition of the bounds and the binary string length for each design variable, along with any GA parameters such as population size.

2.1.4 Genetic Operators

The three genetic operators that compose the core of a GA are selection, crossover, and mutation. These operators simulate genetic evolution; attempting to achieve adequate exploration of the design space and promote the designs that are best adapted to the fitness landscape. The location and sequence of the core operators is illustrated in a flowchart of a simple GA in Figure 2.1. In addition to the three standard operators, there are auxiliary operators that may be implemented to model other genetic phenomena. Additional operators may also be incorporated to improve convergence properties such as encouraging the evolution of multiple species through a niching operator or guaranteeing the survival of the best individuals from one generation to the next using an elitism operator. The genetic operators incorporate random processes and simple rules but, assembled together, form an effective evolutionary search process.

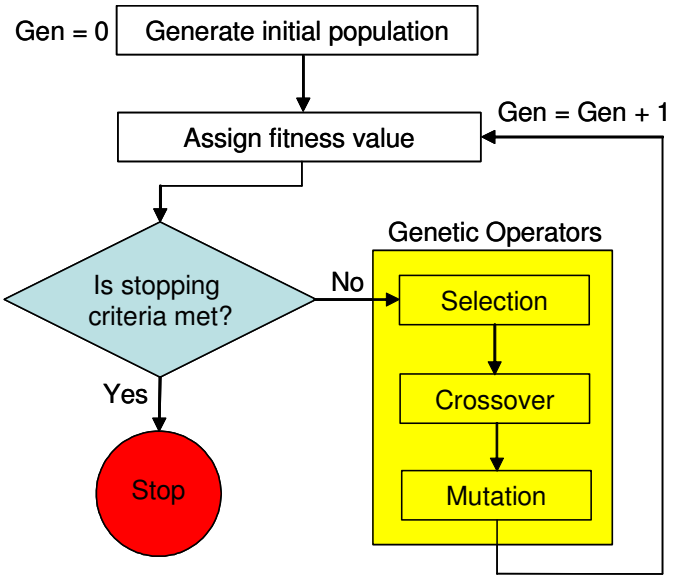

Figure 2.1 Flowchart of a simple genetic algorithm

2.1.4.1 Selection

Selection is the "survival of the fittest" operator in a genetic algorithm. This operator determines which designs from the population will survive to form the 'parents' of the next generation. The selection operator is the mechanism through which the GA establishes which individuals are best adapted to the fitness landscape and should have their genes advanced to future generations. Individuals that are more fit to the design 'environment' will be more likely to survive and pass on their traits. This procedure is analogous to natural selection as described by Darwin in the *Origin of Species* [42].

Several methods of selection exist, but one of the most effective is tournament selection [43]. In tournament selection, two designs (or chromosomes), are selected at random to compete against each other. The fitness values of the competing designs determine which individual wins the tournament and is placed in a parent pool. Those two individuals are then set aside and the tournament is continued until all designs in the population have competed and the winners placed in the parent pool. At this point, the parent pool is half the size of the population. To fill the parent pool, so that it is the same

size as the population, a second round of competitions is conducted. Again, at random, two members of the population are put into competition, and the winning design added to the parent pool. After the second round, the parent pool is the same size as the original population. However, the better adapted members retain a higher probability of placement in the parent pool.

An example of tournament selection for a minimization problem is illustrated in Figure 2.2 using the population defined in Table 2.1. The population size is six, each individual with a chromosome representing its design variables and a corresponding fitness value. As described above, for each of the two rounds, two designs are selected at random for competition and the fitter individual moves on to the parent pool. Note that after the tournament, the two best designs, individuals 3 and 4, have two copies in the parent pool, while the two worst designs, individuals 5 and 1, are not represented.

Table 2.1 Example population with binary chromosome and corresponding fitness

Individual	**Chromosome**	**Fitness**
1	0100101	34.1
2	1101011	19.4
3	0011010	2.7
4	1010011	16.6
5	0011100	80.4
6	1001101	20.5

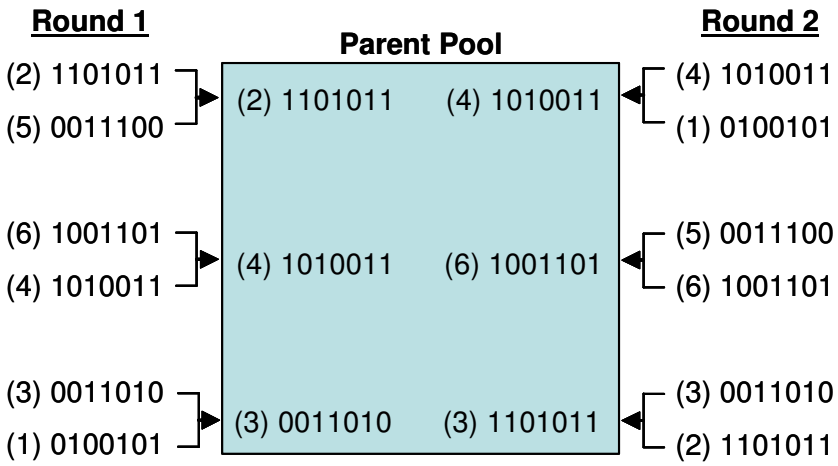

Figure 2.2 Example of tournament selection

All selection strategies ensure that the best individuals in a population are more likely to 'mate'. However, an advantage of tournament selection is the guarantee that the best individual is included in the parent pool twice, while the worst individual is never accepted. This guarantee affords the best design the opportunity to mate twice in the crossover operator, and to pass on its traits (or binary patterns) at a higher rate. The worst individual is eliminated, and cannot produce any offspring.

2.1.4.2 Crossover

Whereas selection determines which designs should reproduce, the crossover operator creates new designs to explore the design space. To form new binary combinations, and thus design characteristics, the crossover operator mates individuals from the parent pool to produce offspring. This breeding process allows the genes from the parent pool to be passed on to a new generation while generating new patterns that may prove to be advantageous. The offspring resulting from crossover receive traits from the parent pool (the high-performing designs of the population), and thus possess an improved probability (i.e., better than random) of containing a new design with a better fitness than the previous generation. If an offspring design is poor, however, it will be eliminated in

the next selection operation. In this way, crossover attempts to exploit the knowledge that has been built into the parents through the evolution of the past generations, while simultaneously exploring uncharted regions of the design space. Combined, selection and crossover provide GAs with the ability to efficiently locate promising areas of the design space.

As is the case with selection, there are several different versions of the crossover operator. However, empirical evidence has demonstrated that uniform crossover is very effective for the discovery of new chromosome patterns by producing diverse offspring [44]. Uniform crossover, like most other crossover methods, begins with a random selection of two parents from the parent pool created by selection. The two parents are then mated to produce two children that will comprise part of the next generation. Once parents have mated, they are discarded and two new parents are selected at random from the parent pool to generate offspring. The process is repeated until a new offspring population is complete. The principal difference between various versions of the crossover operator is the process through which the two parents produce offspring.

In uniform crossover, parents are combined such that each bit location in their chromosome is a crossover point. Every slot in the two children's chromosome is then filled by a virtual coin flip to determine which child should receive its bit from the first parent, and which child should receive its bit from the second parent. For example, each bit location in the chromosome of the first child has a 50% probability of receiving the bit from first parent. If the first child does not recieve its bit from the first parent, it is then seeded by the second parent, while the second child is seeded by the first parent. However, if the two parents contain the same bit at a location on the chromosome, both children will inherit that same bit. This process allows for patterns common to both parents to be carried on to the next generation – the exploitation role of crossover.

An example of uniform crossover is illustrated in Figure 2.3. Individuals 3 and 6 are selected at random from the parent pool in Figure 2.2 to form a pair of parents. The chromosome slots of the children are then determined by a virtual coin flip: heads ("h") implies that Child 1 will receive its bit from Parent 1 and Child 2 will inherit its bit from

Parent 2, and vice versa if the flip is tails ("t"). If both parents have the same bit, no flip is required and both children receive the common bit.

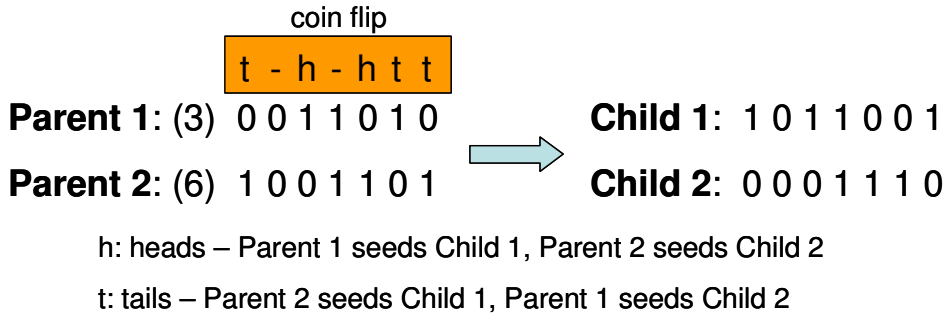

Figure 2.3 Example of uniform crossover

Other versions of the crossover operator do not offer as much explorative capacity as uniform crossover, but can be less disruptive to beneficial binary patterns already in the parent pool [39]. For example, in single-point crossover, one crossover site is selected randomly as the location for a swap of whole sections of the parent's chromosomes. Similarly, in double-point crossover two sites are selected for switching. Both single- and double-point crossover retain more of the structure of the parent chromosomes. For this investigation, only uniform crossover is employed, though the use of other crossover techniques such as single- and double-point crossover deserve further examination.

2.1.4.3 Mutation

The final genetic operator in a traditional scheme is mutation. The mutation operator encourages diversity by slightly altering the chromosomes in the newly created population. Such an operator introduces new variations into the gene pool for a larger exploratory capacity. Mutation typically occurs at a low probability rate, randomly switching a small percentage of the bits in the chromosomes of the population. As an example, consider the chromosome of Child 2 from Figure 2.3: 0001**1**10. If the fifth bit of the chromosome is mutated, the new chromosome becomes 0001**0**10.

Mutation can be implemented in several ways; the simplest of which is a bit-by-bit biased coin flip with a user-specified probability, p_m. If the decision from the biased coin flip is to mutate, the current bit will be changed to its complement. However, this method involves generating a random number for every bit in every chromosome. A less computationally intensive technique is the mutation clock operator [39]. A mutation clock operator uses an exponential distribution to generate the location of the next bit to be mutated. The number of bits to skip from the current location η, is defined by

$$\eta = -p_m \ln(1-\gamma) \tag{2.2}$$

where γ is a random number between 0 and 1. Other mutation schemes that exploit problem specific information can also be implemented.

2.1.4.4 Elite Preservation

An elitism operator is supplementary to the basic simple GA. The purpose of elitism is to guarantee that the best design(s) from previous generations are passed on to future generations. Thus, the best solution in the population cannot degrade from generation to generation. Preserving the previously discovered best design(s) ensures that the GA will not be required to rediscover promising areas of the design space, and it also affords the elite designs more mating opportunities.

Like the other operators, different implementations of elite preservation have been developed, and selection of the most appropriate variation is problem specific. For this application, a specified percentage of the best performing designs from the previous generation replace the same percentage of the worst designs of the current population. The insertion of the elite designs occurs after the fitness evaluation of the current population, but before selection. This approach increases selection pressure on the elites, which can improve efficiency but can also adversely affect diversity. However, uniform crossover can offset the loss of diversity that may be attributed to this formulation of elitism.

2.1.5 Constraints

Equality and inequality constraints can add significant complexity to an optimization problem and several different techniques for the accommodation of constraints in genetic algorithms have been devised [45,46]. The basic genetic operators are not formulated to explicitly manage constraints and, thus, most popular constraint handling methods involve penalizing the fitness value corresponding to infeasible designs [39,47]. However, penalty function methods are not the most appropriate technique for all problems. For problems with extensive constraints and small feasible regions, genetic algorithms incorporating penalty methods can expend many generations attempting to locate only feasible solutions, without progressing toward both feasible and optimal regions. Other methods that preserve feasible over infeasible solutions, or that use a stochastic ranking of feasibility and objective function value, can be better suited and more efficient for some problems [48]. Empirical studies comparing the different methods demonstrate that the best approach is problem dependent and selecting the optimal technique a priori is difficult [49]. Moreover, highly constrained problems can be exceptionally challenging without constraint gradient information to guide the search.

Although not necessarily the optimal approach, penalty function strategies are easy to implement and have wide applicability. For this reason, a penalty function approach is used here. The general statement of a constrained optimization problem is:

$$\begin{aligned} &\text{minimize:} && f(\mathbf{x}) && \\ &\text{subject to:} && g_j(\mathbf{x}) \le 0 && j = 1, m \\ & && h_k(\mathbf{x}) = 0 && k = 1, l \\ & && x_i^L \le x_i \le x_i^U && i = 1, n \end{aligned} \tag{2.3}$$

It is prudent to normalize the constraints to maintain the order of magnitude of various constraints. For example, an inequality constraint can be transformed from

$$\sigma_j(\mathbf{x}) \le \tau_j \tag{2.4}$$

to a normalized, standard form

$$g_j(\mathbf{x}) = \frac{\sigma_j(\mathbf{x})}{\tau_j} - 1 \le 0 \tag{2.5}$$

where $\sigma_j(\mathbf{x})$ is specified function less than or equal to a limit, τ_j. Similarly, an inequality defined as

$$\sigma_j(\mathbf{x}) \ge \tau_j \tag{2.6}$$

takes the form

$$g_j(\mathbf{x}) = 1 - \frac{\sigma_j(\mathbf{x})}{\tau_j} \le 0 \tag{2.7}$$

The equality constraints are written in a similar normalized form. For a penalty method application, the problem is transformed into one that is unconstrained, similar to the exterior penalty method of calculus-based methods. The fitness function, $\varphi(\mathbf{x})$, becomes a combination of the objective function, $f(\mathbf{x})$, and penalty function, $P(\mathbf{x})$:

$$\varphi(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x}) \tag{2.8}$$

Various forms of the penalty function exist allowing different weighting based on the extent of the constraint violation [50]. The exterior quadratic form used by calculus-based methods such as Sequential Unconstrained Minimization Techniques (SUMT), is written

$$P(\mathbf{x}) = \sum_{j=1}^{m} R_j [\max(0, g_j(\mathbf{x}))]^2 + \sum_{k=1}^{n} r_k \left| h_k(\mathbf{x}) \right|^2 \tag{2.9}$$

where R_j and r_k are positive, user-defined penalty parameters. For GAs, $P(\mathbf{x})$ is not required to be continuous and an exterior linear form is available, i.e.,

$$P(\mathbf{x}) = \sum_{j=1}^{m} R_j [\max(0, g_j(\mathbf{x}))] + \sum_{k=1}^{n} r_k \left| h_k(\mathbf{x}) \right| \tag{2.10}$$

Note that the penalty function is evaluated at a positive, nonzero value if the design does not violate any constraints and equals zero if the design is feasible. Other forms such as an exterior step-linear function have also been implemented with success [50].

Both the form of the penalty function and the value of the penalty parameters, R_j and r_k, modify the fitness landscape. With a large penalty, feasible designs possess a strong advantage over infeasible designs regardless of the design's objective function value. Consequently, the number of infeasible designs is reduced more quickly as the evolution progresses. However, a penalty that is too high may distort the landscape so dramatically that artificial local minima are created, complicating the design space. Additionally, infeasible designs may also contain important genes that could be lost if such a strong focus is placed on feasible designs too early in the evolution. Conversely, a weak penalty can weight the infeasible designs too heavily, and locating the feasible regions of the design space becomes more difficult. Thus, it is a critical and nontrivial process to develop the proper penalty parameter and penalty function form for many problems.

To afford additional control over the weighting of infeasible designs in penalty functions, dynamic penalty parameters can be used [51]. The penalty parameters R_j and r_k can either be set as constants (static penalty method), or can vary based on the generation number (dynamic penalty method). For dynamic penalties, the magnitude of the penalty increases with the generation number, assigning a weaker penalty during earlier generations. This approach allows for an exploration of infeasible regions early in the evolution, and then the focus on feasible regions is increased as the generations proceed.

2.1.6 Stopping Criteria

Because the genetic algorithm uses stochastic instead of deterministic transition rules, a proof of convergence for the GA does not exist. Thus, some other method is required to determine when the GA has converged or should stop creating new generations. Three common stopping criteria are:

1. Maximum generation number

2. No fitness value improvement in several generations
3. Population homogeny

Almost all implementations of a GA specify a maximum generation number that terminates the evolution. However, the maximum generation number must be set sufficiently high to ensure that the algorithm evolves long enough to converge if possible. A second safeguard that prevents unnecessary evaluations, is to stop the GA if the fitness value has not improved over several consecutive generations. This cutoff is applied if the algorithm is unable to advance the current best solution. Finally, a measure of the population's homogeny can indicate when exploration has effectively halted because all designs are similar. Bit-string affinity, a measure of the percentage of the bits that are the same in each chromosome, has proven to be a successful method for determining this similarity [52].

2.2 Multiobjective Genetic Algorithm

In engineering applications, consideration of a single objective is frequently not sufficient when selecting a design. These multiple objectives are also often competing or conflicting, such that improvement in one objective will degrade another. In astrodynamics, the two most frequent objectives in interplanetary trajectory design are time-of-flight (TOF) and the final spacecraft mass. A spacecraft with a propulsion system that uses the least control effort will arrive at the target destination with the highest mass (in a fixed initial mass scenario). In general, a trajectory with a longer flight time will possess a higher final mass, whereas a short flight time typically corresponds to a lower final mass. Selecting the trajectory that is best-suited for a mission typically requires a compromise between these objectives. Thus, a multiobjective technique that can simultaneously optimize both time-of-flight and final mass would significantly aid the decision process to emerge with the best final design.

In contrast to single objective optimization, where the goal is a single optimal result, multiobjective optimization techniques generate an entire set of equally optimal solutions. Traditionally, multiobjective optimization has been accomplished by

modifying single objective schemes and executing an optimization routine many times to generate a set of solutions. However, the genetic algorithm's nontraditional, population-based formulation allows the evaluation of an entire set of possible solutions. Therefore, by adapting the fitness function to incorporate all objectives, a multiobjective optimization can be completed in a single run of a genetic algorithm, a distinct advantage over traditional methods. Hence, for a trajectory design problem, both final mass and time-of-flight can be optimized, to produce a set of equally optimal trajectories.

2.2.1 Multiobjective Optimization

In multiobjective optimization, the problem is the minimization of the components in a vector-valued function. The multiobjective optimization problem is formally stated in a general form as

$$\begin{aligned} &\text{minimize}: && \mathbf{f}(\mathbf{x}) \\ &\text{subject to}: && g_j(\mathbf{x}) \le 0 \qquad j = 1, m \\ &&& h_k(\mathbf{x}) = 0 \qquad k = 1, l \\ &&& x_i^L \le x_i \le x_i^U \qquad i = 1, n \end{aligned} \tag{2.11}$$

where **f(x)** is a vector of objectives

$$\mathrm{f(x)} = \begin{Bmatrix} f_1(\mathrm{x}) \\ f_2(\mathrm{x}) \\ \vdots \\ f_{n_{obj}}(\mathrm{x}) \end{Bmatrix}, \tag{2.12}$$

x is a vector of design variables, and n_{obj} is the scalar number of objectives, i.e., the number of rows in **f**(**x**). The objective space, the space in which the objective vector belongs, is n_{obj}-dimensional, whereas in the single objective case the space is one-dimensional. Often in multiobjective optimization, the objectives are not only competing but also coupled, where variables in one objective function appear in the other functions.

In general, because there are multiple design spaces that are different, multiobjective optimization is more complex than single objective optimization. Classical methods, such as the weighted sum approach [53] and the ε-constraint method [54], incorporate single objective strategies to build up a set of optimal solutions. In the weighted sum approach, a single objective is created by summing and weighting the different objectives. Then, a single objective strategy is applied many times with varying weights to generate the set of optimal solution. In the ε-constraint method, a primary objective is selected and all other objectives are treated as hard constraints of user-specified values, ε_l. The values of ε_l are then varied to determine optimal solutions in the objective space. These techniques, however, require many simulations of a single objective method and often necessitate setting values of parameters that can be difficult to determine a priori. Furthermore, these classic techniques are often unable to identify all areas of the optimal objective space. These drawbacks warrant the development of other multiobjective strategies.

2.2.2 Pareto Optimality

The optimal set of solutions in multiobjective optimization is termed the Pareto-optimal set after Vilfredo Pareto, an Italian economist who generalized the concept in 1906 [55]. A solution is Pareto optimal if no improvement in one objective can be accomplished without adversely affecting at least one other objective. In the objective space, the hypersurface (of lower dimension than n_{obj}) that represents all possible Pareto-optimal solutions is defined as the Pareto front (or frontier). In a problem with two competing objectives, the objective space is a plane; and the Pareto front is a curve in the plane. A design that is located along the Pareto front is neither better nor worse than any other solution along the Pareto front. That is, the solutions that compose the Pareto-optimal set are equivalently optimal. Thus, the goal of multiobjective optimization is to generate as many Pareto-optimal solutions as possible to adequately represent the Pareto front, so that sufficient information for a tradeoff decision is available. The Pareto front can be discontinuous as well as concave or convex, and, in general, is not known a priori.

Note, however, that if the objectives are not competing, the Pareto front will be a single point and the problem is effectively a single objective optimization problem.

As an example, consider again the optimization of the competing objectives for an interplanetary trajectory: final mass and time-of-flight. The feasible objective space along with seven designs is illustrated in Figure 2.4. Because final mass is to be maximized and flight time minimized, the Pareto front is located in the upper left region of the feasible objective space. Design 1, $\mathbf{x}_1$, and design 5, $\mathbf{x}_5$, are along the Pareto front (convex in this example), and compose the Pareto-optimal set for this group of designs; all other designs are non-optimal. Though $\mathbf{x}_5$ has a higher final mass than any other design, $\mathbf{x}_1$ has shorter time of flight; neither $\mathbf{x}_1$ nor $\mathbf{x}_5$ is *dominated* by any other design.

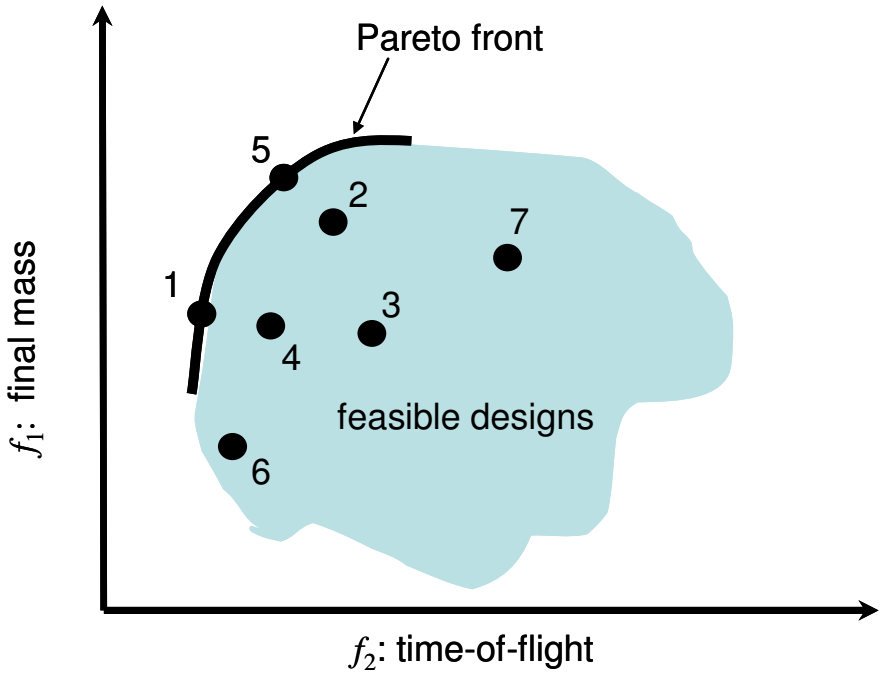

Figure 2.4 Example: Pareto front for trajectory design

2.2.3 Domination

The concept of domination allows for the comparison of a set of designs with multiple objectives. Such a concept is not required for single objective optimization because the value of the objective function is the sole measure of the quality of the design. Recall that $\mathbf{f}(\mathbf{x})$ is the vector of objectives with elements $f_p(\mathbf{x})$. When comparing two multiobjective designs (with all objectives to be minimized), the design $\mathbf{x}_1$ *dominates* design $\mathbf{x}_2$ if:

$$\begin{aligned} &\forall p : f_p(\mathbf{x_1}) \le f_p(\mathbf{x_2}) \qquad p = 1,2,...,n_{obj} \\ &\qquad\qquad\qquad \textit{and} \\ &\exists p : f_p(\mathbf{x_1}) < f_p(\mathbf{x_2}) \qquad p = 1,2,...,n_{obj} \end{aligned} \tag{2.13}$$

That is, $\mathbf{x}_1$ dominates design $\mathbf{x}_2$ if, for all objectives, $\mathbf{x}_1$ is better than or equal to $\mathbf{x}_2$, *and* $\mathbf{x}_1$ outperforms $\mathbf{x}_2$ for at least one objective. Thus, in a direct comparison of two designs, if one design dominates another, the dominating design is superior and "nearer" to the Pareto front. If neither design dominates the other, the designs are non-dominant to each other (analogous to two designs with the same objective value for the single-objective case). Thus, the best designs (with equally good objective vectors) in an arbitrary set of solutions can be distinguished because they are not dominated by any other design in the set; they compose the non-dominated subset. Similarly, the designs that compose the Pareto-optimal set are the non-dominated set associated with the entire feasible space and are located along the Pareto front. However, note that a design can be non-dominated and not be part of the Pareto-optimal set if the entire feasible objective space is not considered.

To elucidate the concept, consider again the trajectory optimization example in Figure 2.4. Systematically comparing all designs yields the results in Table 2.2. Comparing each design with every other design reveals the non-dominated set as well as a hierarchy of non-domination. Comparing $\mathbf{x}_1$ with each of the other designs indicates that $\mathbf{x}_1$ possesses the shortest flight time; thus, no other design can dominate $\mathbf{x}_1$ and it is therefore non-dominated. Correspondingly, $\mathbf{x}_5$ possesses a higher final mass than any other design and is also a member of the non-dominated set. Because both $\mathbf{x}_1$ and $\mathbf{x}_5$ are along the

Pareto front, they are also Pareto optimal. In examining $\mathbf{x}_2$, it is clear that the design is dominated by $\mathbf{x}_5$. However, $\mathbf{x}_2$ is non-dominant to designs $\mathbf{x}_1$, $\mathbf{x}_4$, and $\mathbf{x}_6$, and dominates $\mathbf{x}_3$ and $\mathbf{x}_7$.

Table 2.2 Domination categorization of designs from example trajectory optimization

Design	Dominated by	Dominates	Non-dominant to
$\mathbf{x}_1$	--	$\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_6$	$\mathbf{x}_2$, $\mathbf{x}_5$, $\mathbf{x}_7$
$\mathbf{x}_2$	$\mathbf{x}_5$	$\mathbf{x}_3$, $\mathbf{x}_7$	$\mathbf{x}_1$, $\mathbf{x}_4$, $\mathbf{x}_6$
$\mathbf{x}_3$	$\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_4$, $\mathbf{x}_5$	--	$\mathbf{x}_6$, $\mathbf{x}_7$
$\mathbf{x}_4$	$\mathbf{x}_1$	$\mathbf{x}_3$	$\mathbf{x}_2$, $\mathbf{x}_5$, $\mathbf{x}_6$, $\mathbf{x}_7$
$\mathbf{x}_5$	--	$\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_7$	$\mathbf{x}_1$, $\mathbf{x}_4$, $\mathbf{x}_6$
$\mathbf{x}_6$	$\mathbf{x}_1$	--	$\mathbf{x}_2$, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, $\mathbf{x}_7$
$\mathbf{x}_7$	$\mathbf{x}_2$, $\mathbf{x}_5$	--	$\mathbf{x}_1$, $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_6$

2.2.4 Non-Dominated Sorting

Domination can also be used to categorize each design within a set into a hierarchy of non-dominated levels or fronts. Each different level of non-domination represents a relative "distance" from the Pareto front. The best non-dominated front is closest to the Pareto front and each subsequent front lags further and is, thus, increasingly inferior. Through this sorting, each design is associated with a front that defines the quality of the design relative to the rest of the group. To isolate the various fronts, the designs that belong to the non-dominated subset of the entire group are first identified. These designs are the best in the group, the closest to (or members of) the Pareto front, and are classified as front 1, or F_1. Any design belonging to F_1 is then temporarily set aside and another comparison process determines the next level of non-dominated designs from the remaining population. This non-dominated subset is front 2, or F_2, and the procedure is repeated until the entire population has been sorted into the appropriate level. For

example, return to the multiobjective trajectory optimization example in Figure 2.5. Designs $\mathbf{x}_1$ and $\mathbf{x}_5$ are clearly members of F_1 because they lie along the Pareto front. Designs $\mathbf{x}_1$ and $\mathbf{x}_5$ are then discarded and a second sorting reveals that $\mathbf{x}_2$, $\mathbf{x}_4$, and $\mathbf{x}_6$ are non-dominated and belong to F_2. A final comparison demonstrates that $\mathbf{x}_3$ and $\mathbf{x}_7$ are non-dominant to each other and compose F_3. Note that the members of the best non-dominated front, F_1, will not always be Pareto optimal as is the case here.

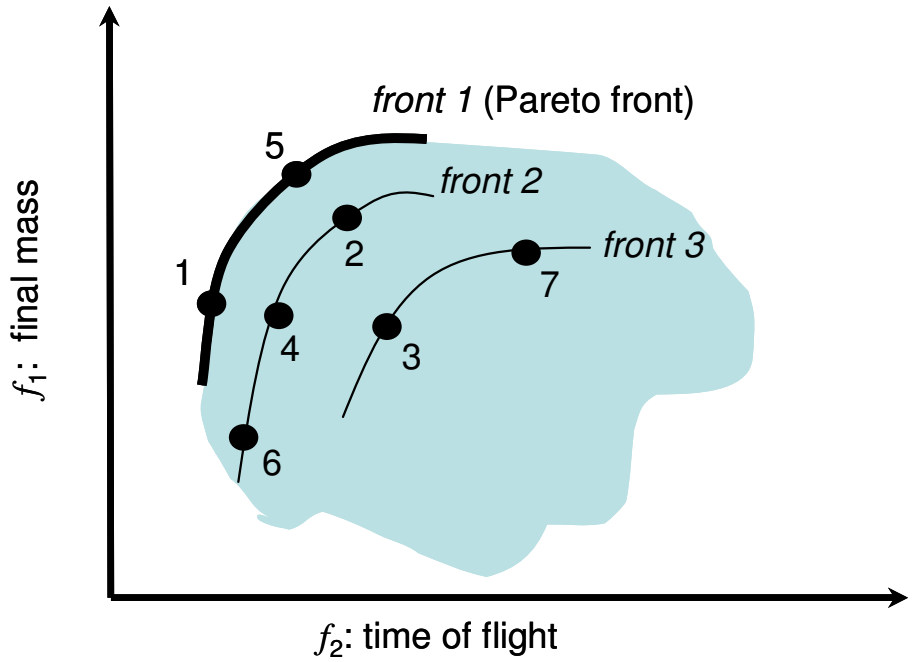

Figure 2.5 Non-dominated fronts for a particular example of designs in multiobjective trajectory optimization

2.2.5 NSGA-II: A Non-Dominated Sorting Genetic Algorithm

The utilization of a population in a genetic algorithm for search and optimization is computationally expensive for single objective optimization, but the population-based approach also affords it the unique ability to operate on many designs simultaneously. Such an approach is advantageous for multiobjective optimization because a proper global Pareto front representation requires many solutions. Thus, instead of many repetitions of a single objective method, one simulation run can develop an entire

population of Pareto optimal solutions. Goldberg realized the potential of a GA for successful application to multiobjective optimization problems; and he outlined a procedure that assigns fitness based on the rank of a design's non-dominated front, determined through non-dominated sorting [39]. Because the designs that are closer to the Pareto front have better fitness values, selection is biased toward the Pareto front. Srinivas and Deb first implemented of this technique with the development of the Non-dominated Sorting Genetic Algorithm (NSGA) [56].

The NSGA scheme attempts to achieve a broad coverage of the Pareto front by both emphasizing the designs that are closest to the Pareto front and preserving diversity. In the NSGA, assigning fitness based on non-dominated sorting (i.e., ranking solutions based on the non-dominated front) ensures movement towards the Pareto front as the generations evolve. To encourage diversity in the population, the NSGA then applies a fitness sharing strategy in which the assigned fitness value is degraded according to the number of designs in the same neighborhood. Typically, the design space neighborhood and not the objective space neighborhood is used. Maintaining diversity is critical to establishing a final population that does not cluster in only limited regions of the Pareto front. After fitness assignment, the population creates a new generation through the three genetic operators: selection, crossover, and mutation. However, it has been demonstrated that tournament selection performs poorly with the sharing strategy. Thus, an alternative selection operator such as proportionate selection is a better option.

The NSGA has been successfully applied to a wide variety of applications; however, the technique is not devoid of pitfalls. Research on the application of the NSGA and other multiobjective evolutionary algorithms indicates that implementing elitism could improve convergence efficiency by ensuring that solutions along or near the Pareto front are not lost [57]. In addition to the lack of an elitism mechanism, the NSGA requires the specification of a sharing parameter to define a neighborhood size. An appropriate value for this parameter is problem specific and experimentation may be required to determine a suitable value. To address these disadvantages, Deb et al. developed a second-generation non-dominated sorting genetic algorithm, the NSGA-II, that employs elitism and eliminates the necessity of a parameter to preserve diversity [53]. The NSGA-II is

especially effective for a large range of difficult multiobjective optimization problems [53,58] and is the technique implemented for this analysis.

The NSGA-II algorithm diverges from the original NSGA by retaining two populations: a parent population, P_t, and an offspring population, Q_t, both of size N. Initially, the parent population is created randomly, consistent with the standard GA, and fitness is assigned through non-dominated sorting. The parent population then produces the first offspring population with the three genetic operators. However, P_t is not discarded; it is combined with Q_t to form a new population, R_t, with size $2N$. Non-dominated sorting then determines the rank of each design in R_t according to its non-dominated front level. After the ranking, a new parent population, P_{t+1}, is created from the best N designs in R_t. The designs in the best non-dominated front receive first priority in filling the available slots in P_{t+1}. Then, if the size of the first non-dominated front is equal or less than N, the designs in the second non-dominated front start filling the remaining available slots. This filling continues with the subsequent non-dominated fronts until P_{t+1} can no longer accommodate an entire non-dominated front from R_t. However, it is unlikely that the last allowed front will fit exactly into the remaining open slots in P_{t+1}. Thus, some mechanism must be employed to determine which designs from the last allowed non-dominated front should be placed into the new parent population. To encourage diversity, preference is given to designs that are less "crowded" in the objective space along the non-dominated front. A sorting based on *crowding distance* differentiates designs in a non-dominated set by favoring designs in less congested areas of the front.

To define a crowding distance it is useful to impose some geometric structure onto a front. The crowding distance, d_i, associated with a design is the sum of the average distance (in objective space) of the two surrounding designs of each objective along a non-dominated front, F. The crowding distance provides a measure of the density of the designs along the front at the location of the i^{th} design by estimating the perimeter of the hyperrectangle formed with the two adjacent designs of each objective as vertices. For a problem with two objectives, such as the trajectory design scenario described previously, the adjacent designs form a rectangle. This scenario is illustrated in Fig. 2.6, where a

non-dominated set and the rectangle formed by the two adjacent members of design *i* are depicted. In general, as the number of objectives increases, the dimension of the geometric structure increases, from a rectangle for two objectives, to a cuboid for three objectives, and then a hyperrectangle. Note, however, that the same two designs, *i*+1 and *i*-1 are not necessarily neighbors for all objectives as is portrayed in Fig. 2.6 for a two objective case.

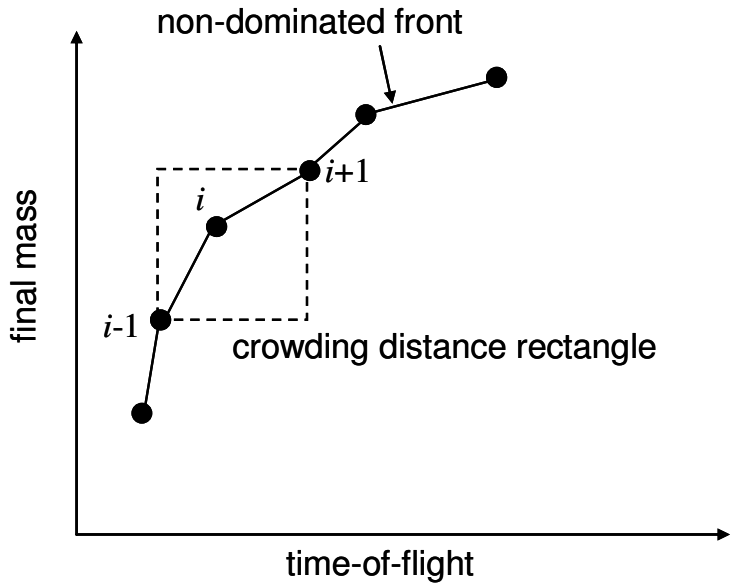

Figure 2.6 Example of crowding distance rectangle formed by adjacent designs in the trajectory design scenario with two objectives (adapted from Deb [53])

A crowding distance is assigned for each member of the non-dominated set of size *s*. The smaller the value of d_i, the less distance separates the neighboring designs from design *i* along the front. A high d_i value indicates that the individual is far from the other designs in the front and is preferred to encourage diversity. Because the designs on the boundaries of the non-dominated front represent extremes in terms of the objectives, and only one adjacent neighbors exists (in contrast to two adjacent neighbors), these boundary designs receive a large crowding distance to ensure their precedence. The crowding distance procedure, as formulated by Deb, is summarized in Fig. 2.7.

Crowding Distance Assignment:

1. Call the number of solutions in F as $s = |F|$. For each design in the set, first assign $d_i = 0$.

2. For each objective function $p = 1,2,\ldots,n_{obj}$, sort the set in worse order of f_p (or find the sorted indices vector: I^p=sort(f_p,>).

3. For $p = 1,2,\ldots, n_{obj}$, assign a large distance to the boundary solutions, and for all other solutions $j = 2$ to (s-1) assign:

$$d_{I_j^p} = d_{I_j^p} + \frac{f_p(I_{j+1}^p) - f_p(I_{j-1}^p)}{f_p^{\max} - f_p^{\max}}, \tag{2.14}$$

where I_j^p represents the index of the j^{th} member of the sorted list corresponding to objective p.

Figure 2.7 Crowding distance assignment procedure for the NSGA-II algorithm (adapted from Deb [53])

The crowding mechanism for completing the last slots in the population P_{t+1} becomes critical to developing a sufficient spread of designs along the Pareto front in the later generations of the NSGA-II algorithm when many designs are already Pareto optimal. In early generations, there are many smaller groups of non-dominated sets and only a few designs in lower-ranking sets require sorting in terms of their crowding distance. In the final generations, however, the best non-dominated front in R_t may, in fact, include more than the allotted N designs and only the least crowded individuals in the set are included in the new parent population. Once created, P_{t+1} moves to the genetic operators, which generate a new, and hopefully more optimal, offspring population, Q_{t+1}.

A second key difference in the NSGA-II occurs in the selection operator, where the preservation of further diversity is enabled. Before selection, each design in P_{t+1} is assigned a fitness value equal to the rank of the corresponding non-dominated. That is, a design along the best non-dominated front is assigned a fitness of 1. Then, the fitness of any design along the second non-dominated front is 2, and so on. However, the NSGA-II algorithm also distinguishes between designs with the same fitness by sorting each front

in the population P_{t+1} in terms of crowding distance. Differentiating designs via the crowding distance metric allows for application of the 'crowded tournament' selection operator and avoids the necessity of specifying the sharing parameter as required by the NSGA. The crowded tournament operator is structured in a manner that is similar to standard tournament selection, but compensates for the possibility of competition between two designs with the same fitness value. The crowded tournament operator compares two designs at a time and, first, evaluates the fitness value, which are equivalent to the ranking of the non-dominated front associated with the designs. If the designs belong to different fronts and, thus, have a different fitness, the design belonging to the better front wins the competition. However, when the fitness values of two competing designs are equal, the design with a larger d_i is selected for promotion to the parent pool. In this way, the crowded tournament selection operator further promotes diversity. After selection, the members of the parent pool are mated in the crossover operator and the resulting children are passed to the mutation operator, which completes the creation of the offspring population.

The NSGA-II strategy of combining both parent and offspring populations for comparison guarantees that Pareto-optimal solutions in the parent population are not lost and survive to become parents in the next generation. This elite preservation yields improved convergence properties and eliminates the addition of parameters that require specification. Furthermore, the NSGA-II includes a strategy to develop a wide spread set of solutions along the Pareto front by affording preference to less crowded designs in two different locations in the algorithm. A summary of the NSGA-II procedure is outlined in Figure 2.8.

Non-Dominated Sorting Genetic Algorithm II:

1. Generate initial population, P_1 (N designs)

2. Use non-dominated sorting to assign fitness to each non-dominated front
   - designs in *front* 1 assigned a fitness of 1, designs in *front* 2 assigned a fitness of 2, etc.

3. Apply GA operators to generate offspring population, Q_1
   - crowded tournament selection, crossover, mutation

4. Combine parent and offspring population into R_t
   - population size is 2N

5. Employ non-dominated sorting on combined population, R_t

6. Assign measure of crowding distance to each solution in R_t

7. Fill N slots with best designs from combined population to create new parent population
   - final slots determined by crowding distance

8. Apply GA operators to develop new offspring population
   - crowded tournament selection: 2 designs with the same fitness value employ the crowding distance measure to prioritize

9. Return to step 4 with parent and new offspring population unless stopping criteria is met

Figure 2.8 NSGA-II algorithm outline

2.2.6 NSGA-II Verification

The NSGA-II is structured for a binary encoding of the design variables and implemented in MATLAB®. Crowded tournament selection, uniform crossover, and clock mutation are applied as the three genetic operators. To verify to the MATLAB code, several test problems with known Pareto fronts were selected for comparison with Deb et al., the originators of the NSGA-II [59]. The code is executed for 250 generations

with a population size of 100. At completion, the final population is compared against the known Pareto front and the results from Deb et al. [59]. The mutation rate for the test cases is set consistent with the empirical guidelines in Williams and Crossley [60], that is

$$p_m = \frac{l+1}{2Nl} \tag{2.15}$$

where l is the chromosome length and N is the population size. Each design variable for the test cases is represented in terms of a binary encoding using 30 bits. The mutation rate guideline and the use of uniform crossover, as opposed to single-point crossover, differ from the NSGA-II as it is applied in Deb et al. [59]. Thus, differences in performance are possible. Only unconstrained problems with two objectives are assessed.

Test Problem 1

The first problem is from Schaffer [61] and tests the ability of a multiobjective algorithm to generate a discontinuous Pareto front:

$$\begin{aligned} &\text{minimize:} \quad f_1(x) = \begin{cases} -x & \text{if } x \le 1 \\ x-2 & \text{if } 1 < x \le 3 \\ 4-x & \text{if } 3 < x \le 4 \\ x-4 & \text{if } x > 4 \end{cases} \\ &\text{minimize:} \quad f_2(x) = (x-5)^2 \\ &\text{subject to:} \quad -5 \le x \le 10 \end{aligned} \tag{2.16}$$

This problem consists of only one design variable but it is a popular benchmark.

As implemented, the NSGA-II algorithm quickly develops the Pareto front by the sixth generation as is apparent in Fig. 2.9. Then, in the remaining generations, an even spread of solutions is established along the front. This corresponds well with the results from Deb et al. [59].

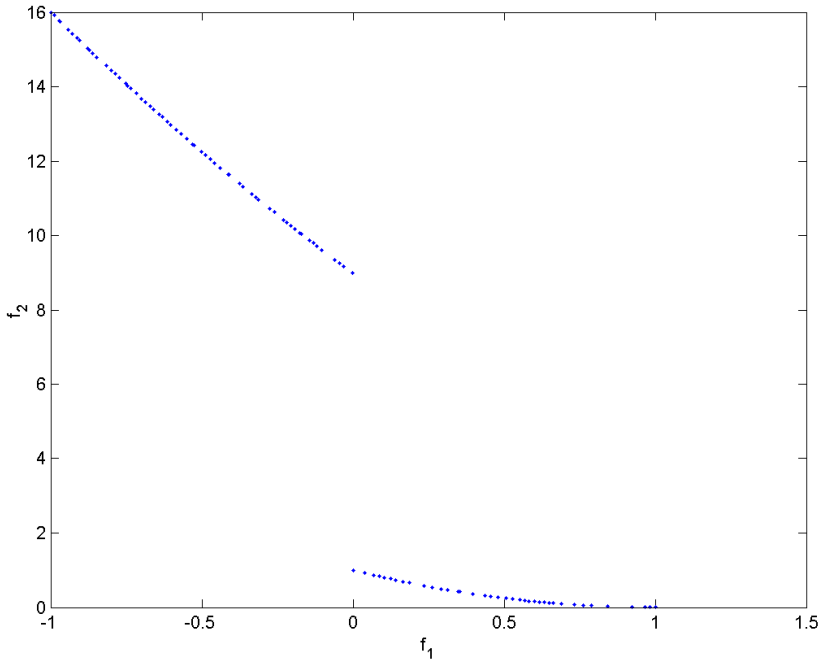

Figure 2.9 Population at 6th generation for test problem 1

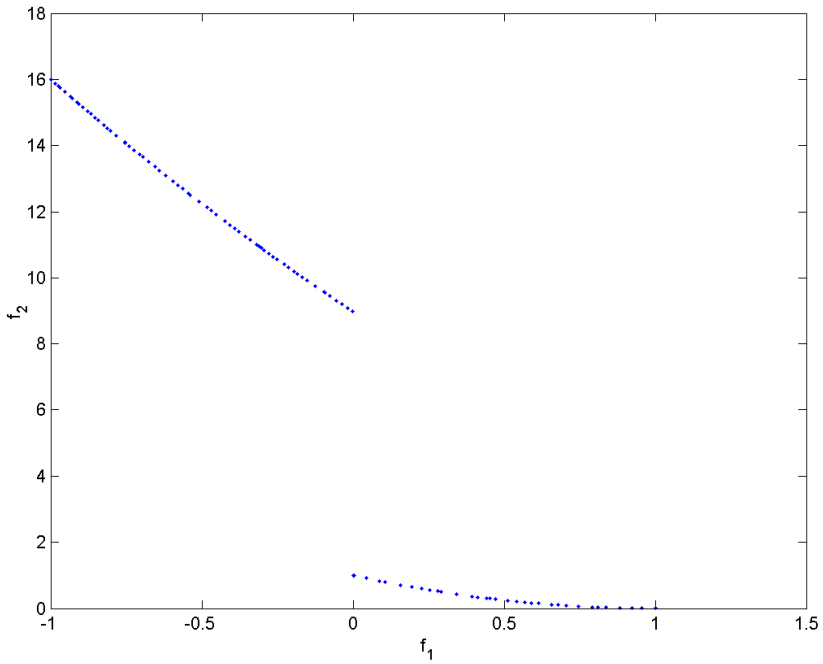

Figure 2.10 Population at 250th generation for test problem 1

Test Problem 2

A problem developed by Kursawe [62] is also investigated by Deb et al.:

$$\begin{aligned} &\text{minimize}: \quad f_1(x) = \sum_{i=1}^{2}\left(-10e^{-0.2\sqrt{x_i^2+x_{i+1}^2}}\right) \\ &\text{minimize}: \quad f_2(x) = \sum_{i=1}^{3}\left(|x_i|^{0.8} + 5\sin(x_i^3)\right) \\ &\text{subject to}: \quad -5 \le x_i \le 5, \quad i = 1,2,3 \end{aligned} \tag{2.17}$$

This test problem is more complex, with three design variables and four disjointed Pareto front regions. The algorithm as applied to this test problem, successfully identifies all four regions as well as an expansive coverage of the Pareto front. The performance characteristics and results of the NSGA-II for this case are similar to those by Deb et al. [59] as illustrated in Fig. 2.11.

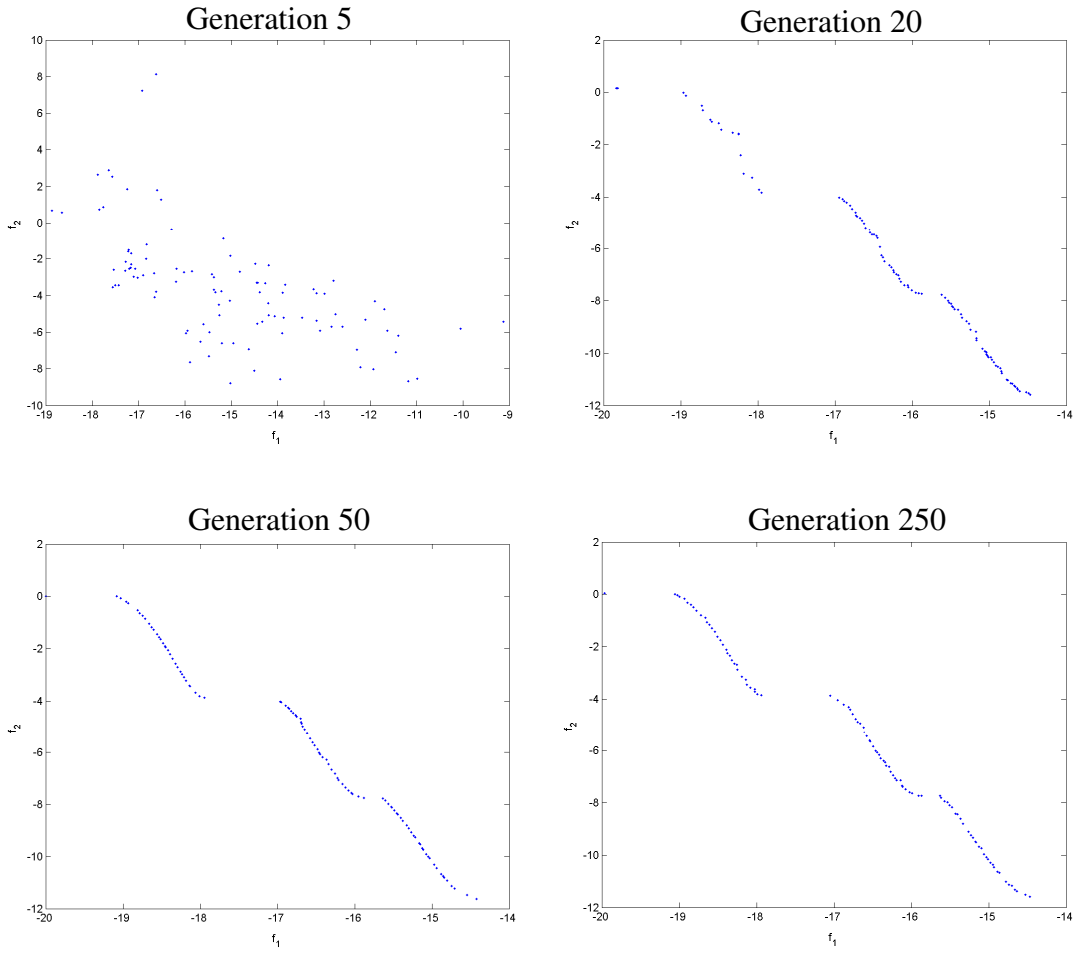

Figure 2.11 Population at generations 5, 20, 50, and 250 in test problem 2

Test Problem 3

Zitzler et al. developed a suite of test problems designed to be difficult for multiobjective evolutionary algorithms [57]. The problems include complex parameter interactions with nonlinear functions to create discontinuous Pareto fronts of various shapes. One of the more difficult problems is formulated as follows:

$$
\begin{aligned}
&\text{minimize}: \quad f_1(\mathbf{x}) = x_1 \\
&\text{minimize}: \quad f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)\right] \\
&\qquad\qquad \text{where } g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{30} x_i\right) / 29 \\
&\text{subject to}: \quad 0 \le x_i \le 1, \quad i = 1,2,...,30
\end{aligned}
\tag{2.18}
$$

In addition to 30 design variables, the Pareto front for this test case is discontinuous and comprises five distinct regions complicating the optimization process.

The simulation of the code delivers all five regions of Pareto front with a distribution of the population uniformly along the front; such a result is accomplished by the 250th generation as evidenced by the plots in Fig. 2.12. However, not all members in the final population are Pareto-optimal and more generations are required to place every design along the Pareto front. The performance of the NSGA-II implementation is consistent with the results from Deb et al.

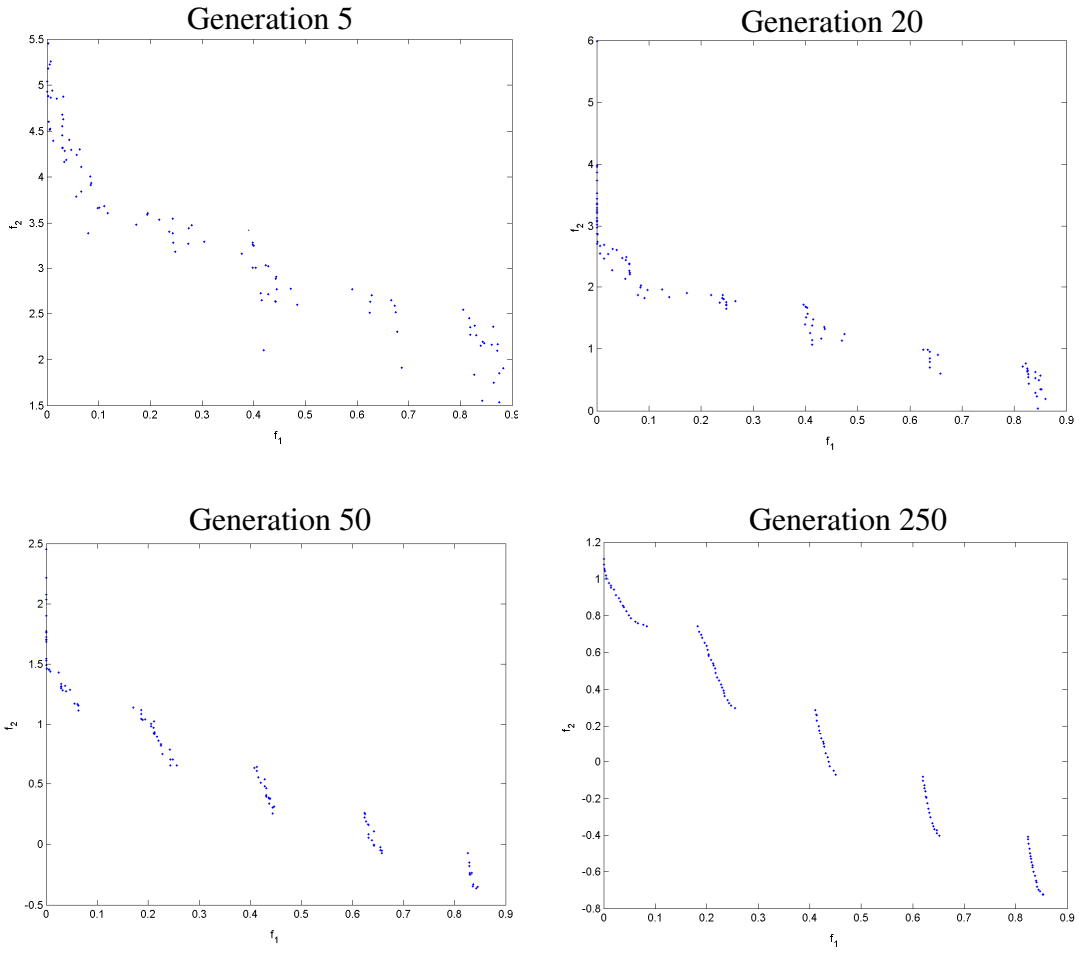

Figure 2.12 Population at generations 5, 20, 50, and 250 in test problem 3

2.3 GALLOP: A Direct Low-Thrust Trajectory Optimization Program

To address the difficulties of optimizing interplanetary low-thrust gravity-assist trajectories, Sims and Flanagan [63] devised a direct method that is the framework for an efficient and robust local optimization scheme. This scheme has been incorporated into two independently created software packages: Gravity-Assist Low-thrust Local Optimization Program (GALLOP) developed by McConaghy et al. at Purdue University [64,65]; and Mission Analysis Low-thrust Optimization (MALTO) constructed at the Jet

Propulsion Laboratory (JPL) [66]. Both programs have been utilized in a number of preliminary solar electric propulsion (SEP) and nuclear electric propulsion (NEP) trajectory design studies [67-69], demonstrating their capability to locally optimize complex low-thrust trajectories with multiple gravity assists.

The structure of the Sims-Flanagan trajectory model affords both GALLOP and MALTO a reduced sensitivity to the initial guess (corresponding to physical variables) when compared to indirect methods. Furthermore, the two programs do not require an initial guess for the non-physical costates (Lagrange multipliers); and the use of analytic derivatives allows for efficient execution. However, the benefits of robustness and efficiency come at the expense of a high dimensionality and limited accuracy due to a simplified trajectory model. Thus, the trajectory produced from GALLOP or MALTO, albeit a generally good approximation [67], must be refined in a higher fidelity model for use on an actual mission.

For application in this investigation, GALLOP is incorporated as the local optimization component of the hybrid approach. The models and the optimization procedure employed by GALLOP are introduced in the following section. Further details are available in McConaghy's [64] and the GALLOP User's Guide [70].

2.3.1 Trajectory Model

The trajectory model in the Sims-Flanagan direct approach models facilitates rapid convergence and minimizes sensitivities. To accommodate the complexities of multiple gravity assists, trajectories in GALLOP are partitioned into independent legs, which are bounded by gravitational bodies, or control nodes, as illustrated in Figure 2.13. Along each leg, a match point is defined at a particular fraction of the leg's duration (often the midpoint of the leg). The trajectory is propagated forward in time from the originating control node of the leg to the match point, and backward in time from the terminating control node to the match point. If a mass, position, or velocity discontinuity exists at the match point, the trajectory arc is infeasible and the optimizer attempts to reduce these discrepancies to produce a feasible trajectory. This multiple shooting strategy decreases

the sensitivities that are incurred by propagation only forward in time from the start of the trajectory. That is, nonlinearities are more readily accommodated, and the problem of small errors, from the initial guess, propagating into large changes by the end of the trajectory is avoided, increasing the radius of convergence of this formulation. This type of targeting bears similarities to that described by Byrnes and Bright and employed in JPL's CATO program [71].

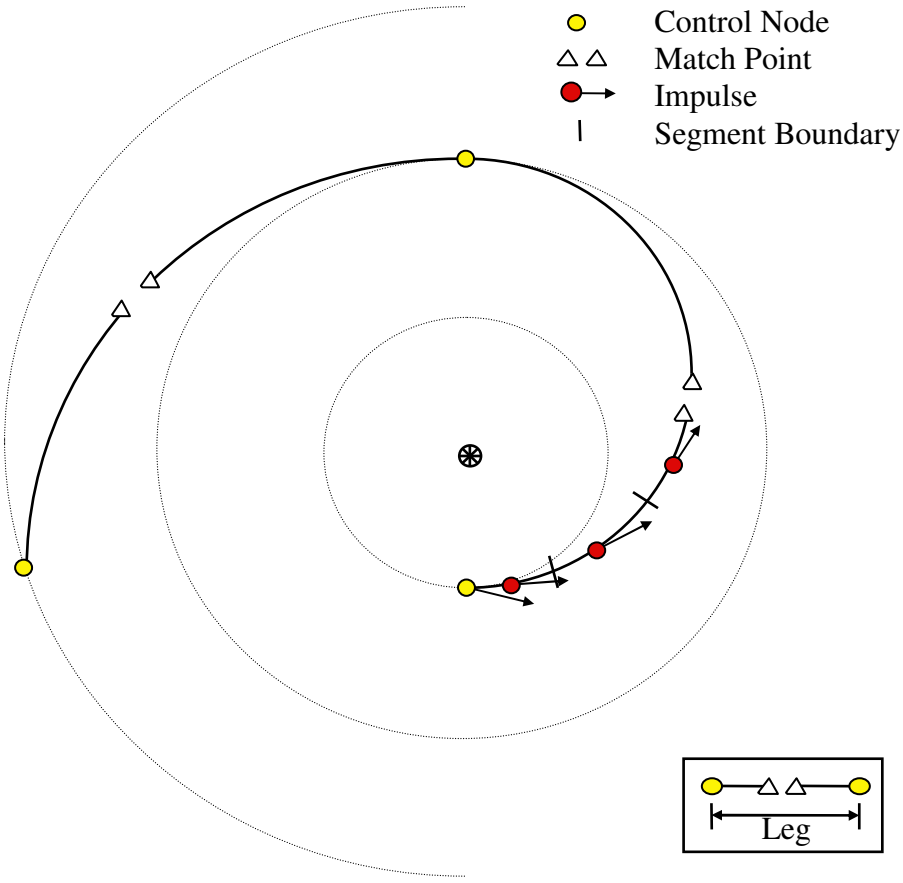

Figure 2.13 GALLOP's trajectory structure (from Sims and Flanagan [62])

A fundamental simplifying assumption is incorporated in GALLOP's modeling of the spacecraft thrust. In contrast to indirect methods, which generate a continuous thrust profile, GALLOP discretizes the continuous thrust arc to create a finite-dimensional problem. Each leg is divided into a specified number of segments and the continuous thrust is approximated by an impulsive ΔV at the midpoint, in time, of each segment. Thus, continuous thrusting is assumed along the entire segment if the ΔV magnitude is greater than zero. Accordingly, the more segments that are specified per leg, the better the thrust approximation. The two sections composing the leg (one before and one after the match point) each comprise segments with equal duration, but the duration for each set of segments may be different because the match point is not necessarily defined at the leg midpoint. The maximum magnitude of the ΔV on each segment is constrained by the capability of the engine over the time interval of the segment, that is:

$$\Delta V_{\max} = \frac{T}{\dot{m}} \ln\left(\frac{m_0}{m_0 - \dot{m}\kappa t} \right) \tag{2.19}$$

Equation 2.19 is based on Tsiolkovsky's rocket equation [72], where T is the thrust on the segment, $\dot{m}$ is mass flow rate, m_0 is the initial mass of the segment, κ is the duty cycle (indicating how long the engine is thrusting when on), and t is the time duration corresponding to the segment.

A two-body gravitational model is assumed in GALLOP, with the Sun as the primary body for interplanetary trajectories. Thus, each interval between control nodes, impulses, and match points is a conic arc, and the state is propagated analytically, significantly reducing computational time. The mass and the velocity of the spacecraft undergo an instantaneous change at each impulse if the ΔV associated with the segment is not zero (position is continuous through the impulses). The two-body approximation is sufficiently accurate given the assumptions in the thrust model, and the large disparity between the gravitational influence on the spacecraft from the Sun and the other bodies during the majority of an interplanetary trajectory. In GALLOP, the user may select either a rendezvous or a flyby for each intermediate or final control node. A rendezvous assumes that the spacecraft matches the position and velocity of the encounter body.

Additionally, if a rendezvous is specified for an intermediate body, a minimum stay time at that body must also be defined. For a flyby, a gravity assist is modeled as an instantaneous rotation of the relative velocity vector, $\mathbf{v}_\infty$, at the control node. The turning of the $\mathbf{v}_\infty$ is limited by the mass of the flyby body and the flyby altitude. In addition, the magnitude of the arrival and departure $\mathbf{v}_\infty$ is constrained to be the equal for this type of encounter.

2.3.2 Launch Vehicle Model

The GALLOP software package incorporates the launch vehicle model developed by Sauer for JPL's SEPTOP program [36]. The user must specify eight launch vehicle parameters that define the relationship between the launch magnitude of the $\mathbf{v}_\infty$ vector and the initial departure mass at the first control node. It is assumed that the payload and upper stage originate in a circular orbit about the Earth at a radius r_{p0}. From this parking orbit, an upper stage burn inserts the spacecraft onto a hyperbolic escape orbit at a specific v_∞ magnitude. Applying the conservation of energy, the ΔV required to reach the necessary relative escape velocity is calculated:

$$\Delta V_c = \sqrt{\frac{2\mu_E}{r_{p0}} + v_\infty^2} - \sqrt{\frac{\mu_E}{r_{p0}}} \tag{2.20}$$

where μ_E is the Earth gravitational parameter. The rocket equation is then applied to determine the mass at the end of the burn, m_0^*. This mass value, the launch vehicle capability, includes the payload mass as well as the upper stage structural mass and the payload adapter mass.

Four parameters, ξ_1 through ξ_4, define the launch vehicle performance characteristics, and their values influence the method applied to determine m_0^*. Each of these parameters is described in Table 2.3. For the launch vehicles used in this study, there is no upper stage propellant mass. Thus, ξ_4 is zero, and the launch vehicle capability is found by manipulating the rocket equation:

$$m_0^* = \xi_1 \exp\left(\frac{-\Delta V_c}{\xi_3}\right) - \xi_2 \,. \tag{2.21}$$

For further details and the derivation of m_0^* for nonzero values of ξ_4 see McConaghy [64].

Table 2.3 Description of launch vehicle model parameters

Parameter	Description
ξ_1	Total initial mass (structure, propellant, adapter, payload) [kg]
ξ_2	Upper stage structural mass [kg]
ξ_3	Propellant exhaust velocity (=gI_{sp}) [km/s]
ξ_4	Upper stage of propellant mass [kg]

After the launch vehicle capability is determined, it is adjusted by the user-defined launch vehicle contingency parameter, k_{lv}. This safety factor is applied to accommodate any unexpected variations during launch, providing a conservative estimate on the vehicle's capability. The value of the launch vehicle contingency parameter is a specified percentage of the delivered mass that is set aside as margin, thus the launch vehicle capability is multiplied by (1 - k_{lv}). Additionally, because the structure of the adapter connecting the payload to the launch vehicle is no longer necessary, the mass corresponding to the adapter is discarded. Hence, to establish the payload mass m_0 (the initial mass at the first control node), the mass of the adapter, m_{adp}, is subtracted from the adjusted launch vehicle capability:

$$m_0 = (1 - k_{lv}) m_0^* - m_{adp} \tag{2.22}$$

However, in some launch scenarios, the mass of the adapter is not fixed and is instead modeled as a percentage k_{adp} of the payload mass. In this scenario, the initial mass is calculated by:

$$m_0 = \frac{(1-k_{lv})m_0^*}{(1+k_{adp})} \tag{2.23}$$

Note that, in GALLOP, a value must be specified for both adapter mass parameters; therefore, either m_{adp} or k_{adp} must be equal to zero, though both can be zero if an adapter is not modeled.

2.3.3 Engine Model

To model the thrust and mass flow rate generated by a single low-thrust engine, GALLOP incorporates polynomial functions based on the power available to the spacecraft. The thrust magnitude, T, is modeled as

$$T = ct_1 + ct_2 P_a + ct_3 P_a^{\,2} + ct_4 P_a^{\,3} + ct_5 P_a^{\,4} \tag{2.24}$$

where P_a is the input power and ct_i are user-defined coefficients based on a particular engine's capability. The units for the five thrust coefficients differ, and are defined: ct_1 [N], ct_2 [N/kW], ct_3 [N/kW^2], etc. Similarly, the propellant mass flow rate is calculated using the polynomial

$$\dot{m} = cm_1 + cm_2 P_a + cm_3 P_a^{\,2} + cm_4 P_a^{\,3} + cm_5 P_a^{\,4} \tag{2.25}$$

where cm_i are also coefficients supplied by the user. The values of these coefficients are associated with the particular engine. The units for cm_i emerge in a pattern similar to the previous set of coefficients: cm_1 [kg/s], cm_2 [(kg/s)/kW], cm_3 [(kg/s)/kW^2], etc.

The power available to the spacecraft may originate from solar arrays (SEP missions) or from a nuclear reactor (NEP applications). For a SEP spacecraft, the minimum power required to the start the engine, P_{min}, and the maximum usable power, P_{max}, must both be specified. However, for NEP models the input power is assumed to be constant, with thrust and mass flow rate calculated from

$$T = \frac{2\eta P_a}{g_0 I_{sp}} \tag{2.26}$$

and

$$\dot{m} = \frac{T}{g_0 I_{sp}} \tag{2.27}$$

respectively, where g_0 is the standard acceleration due to gravity, and η is an engine efficiency factor. Therefore, the thrust and mass flow rate remain constant and ct_1 and cm_1 are the only nonzero coefficients in GALLOP's model for a NEP engine.

In this investigation, only a single-engine spacecraft is considered. However, GALLOP includes the option to model spacecraft with multiple low-thrust engines. When there is more than one engine, the power available can be distributed to the engines in various ways for each thrusting segment – each engine can be either on or off and each engine can receive a different power input. Because of the number of possible combinations of engines and operating conditions over all the segments along an entire trajectory for a multiple-engine model, GALLOP incorporates a set of heuristic rules to determine how best to allocate the available power. McConaghy [64] describes the additional parameters that are required for the multiple-engine model. The additional capability is not exploited in this analysis.

2.3.4 Solar Array Model

Similar the launch vehicle model, GALLOP incorporates a solar array model based on the representation developed for SEPTOP. The solar array model determines the power available to the spacecraft as a function of radial distance from the Sun and time from launch. The power generated by the solar array increases with decreasing distance from the Sun, that is

$$P_{gen} = P_0 \varepsilon P_{rel} \tag{2.28}$$

where P_0 is the reference power for the solar array at one astronomical unit (AU), ε is a time-dependent efficiency factor, and P_{rel} is a nondimensional relative power based on the spacecraft's distance from the Sun. The efficiency factor ε is a number between zero and one, allowing for solar array degradation as the time from launch, t, increases. Thus, ε is evaluated from the following expression

$$\varepsilon = \beta_1 + \beta_2 e^{\beta_3 t} + \beta_4 t \tag{2.29}$$

where the parameters β_1 through β_4 are defined by the user. Parameters β_1 and β_2 are dimensionless, while the units for both β_3 and β_4 are s^{-1}. The relative power P_{rel} is a function of the solar array distance from the Sun and is calculated as follows:

$$P_{rel} = \frac{\alpha_P}{r_{sun}^2}\left(\frac{\gamma_1 + \dfrac{\gamma_2}{r_{sun}} + \dfrac{\gamma_3}{r_{sun}^2}}{1 + \gamma_4 r_{sun} + \gamma_5 r_{sun}^2}\right) \tag{2.30}$$

where r_{sun} is the spacecraft's distance from the Sun, and α_P is a characteristic quantity equal to one AU^2 that contributes to the nondimensionalization of P_{rel}. The bracketed term in Equation 2.30 applies the user-defined parameters γ_1 through γ_5 to model the effect of temperature on the array's efficiency, and is equal to one when r_{sun} = 1 AU. The units for γ_i are as follows: γ_1 is dimensionless, γ_2 [AU], γ_3 [AU^2], γ_4 [AU^{-1}], γ_5 [AU^{-2}].

Because the spacecraft requires power for operations other than thrusting the engines may not receive all of the power generated by the solar arrays. As a result, the power available to the engines is

$$P_a = P_{gen} - P_{sc} \tag{2.31}$$

where P_{sc} is the power required for any operations other than engine thrust.
However, if P_{sc} is greater than P_{gen}, no power is available for the engines.

2.3.5 Optimization Problem

The formulation of GALLOP's trajectory model represents a constrained, nonlinear optimization problem, stated in general form as

$$\begin{aligned} &\text{minimize}: && f(\mathbf{x}) \\ &\text{subject to}: && g_j(\mathbf{x}) \le 0 \\ & && h_k(\mathbf{x}) = 0 \\ & && x_{\mathrm{i}}^{\mathrm{L}} \le x_i \le x_i^U \end{aligned} \tag{2.32}$$

Both inequality and equality constraints (g_j and h_k respectively) on the independent design variables, $\mathbf{x}$, result from the Sims-Flanagan formulation. Additionally, a lower and upper bound for each of the independent variable must be specified. The goal of the optimizer is to adjust the independent variables from an initial guess such that a trajectory is feasible (i.e., constraints are within an acceptable tolerance), and optimal (the Kuhn-Tucker conditions are met [41]). The objective for this analysis is the maximization of the final spacecraft mass, such that $f(\mathbf{x}) = -m_f$; though minimization of the time-of-flight is also available in GALLOP.

2.3.6 Independent Variables

The structure of the trajectory in GALLOP requires a significant number of independent variables, or optimization variables, due to the discretization of the continuous thrust. There are also several optimization variables associated with each control node. For the initial control node, or launch body, the independent variables are defined as the departure epoch, the initial mass if a launch vehicle has not been specified, and the departure relative velocity vector. If a launch vehicle is used, the initial mass of the trajectory is determined by the magnitude of the departure $\mathbf{v}_\infty$. Correspondingly, at an intermediate control node there are, in general, time, mass, and relative velocity independent variables associated with both the arrival and departure of the spacecraft at the body. For a flyby encounter, the incoming and outgoing relative velocity will be

equal in magnitude but different in direction. As a result, the independent variables for the relative velocity are typically represented by the three components of the arrival velocity and a flyby altitude with an associated B-plane angle. The B-plane is defined as the plane perpendicular to the asymptote of the incoming relative velocity vector and that passes through the encounter body's center of gravity. The B-plane angle is the angle in the plane between (a) the line formed by the intersection of the B-plane with the ecliptic plane *and* (b) the line extending from the encounter body's center of gravity to the point where the asymptote intersects the B-plane. Only two (instead of three) variables are required to represent the outgoing relative velocity because its magnitude must be equal to the magnitude of the incoming relative velocity vector. The arrival and departure variables for mass and time do not differ for a flyby body. Contrastingly, for a rendezvous at an intermediate body, the departure and arrival times can be different in addition to dissimilar relative velocity magnitudes. At the final body, only the arrival mass (the optimization objective), arrival time, and incoming $\mathbf{v}_\infty$ are optimization variables.

The three components of the thrust vector on each segment over the entire trajectory comprise the majority of the independent design variables. The number of segments that are required to adequately model the continuous low-thrust arc adequately depends on several factors, including the duration of the leg, the number of solar revolutions, and the thrust magnitude that can be delivered by the enginge. For example, a leg from Earth to Venus would typically require 20 to 40 segments. For a leg from Earth to Mercury, however, over 100 segments may be necessary. Thus, from 60 to 120 independent variables are required to represent the thrust for an Earth-to-Venus leg, and over 300 variables may be necessary for an Earth-to-Mars leg. For multiple gravity-assist trajectories with several legs, the number of independent variables and complexity of the optimization problem quickly increases with each additional intermediate body. The independent variables associated with GALLOP are summarized in Table 2.4.

Table 2.4 Description of independent variables in GALLOP formulation

	Independent Variables	Notes
Initial Body	t_0, $\mathbf{v}_{\infty,\text{dep}}$, m_0	m_0 is not included if a launch vehicle defined
Intermediate Body	t_{arr}, t_{dep}, $\mathbf{v}_{\infty,\text{arr}}$, $\mathbf{v}_{\infty,\text{dep}}$, m_{arr}, m_{dep}	flyby altitude and B-plane angle can replace $\mathbf{v}_{\infty,\text{dep}}$ for a flyby
Final Body	t_f, $\mathbf{v}_{\infty,\text{arr}}$, m_f	$-m_f$ is the objective function to be minimized
Thrust	$\boldsymbol{T}_i$	i = 1,.., total number of segments

2.3.7 Constraints

Both inequality and equality constraints exist in GALLOP's formulation. Moreover, those constraints can be either linear or nonlinear. Linear constraints can be imposed to limit the time-of-flight for each leg or the duration of the entire trajectory. However, the critical constraints enforcing mass, position, and velocity continuity at the match point are nonlinear. These equality constraints force the forward propagated section of a leg and the section that is propagated backwards in time to arrive at the same state. The other set of nonlinear constraints is comprised of the upper limits on the magnitude of ΔV impulses at each segment midpoint. The complete set of nonlinear constraints must be satisfied for a physically feasible trajectory

There are also constraints associated with an intermediate body encounter. If the intermediate body is a flyby, the magnitude of the relative velocity vectors must be equal. However, if the body is a rendezvous, the incoming relative velocity magnitude must be zero.

2.3.8 The Optimization Process

To perform the optimization, GALLOP incorporates the nonlinear optimization software SNOPT (Sparse Nonlinear OPTimizer) developed by Gill [73]. The optimizer, SNOPT, is based on a sequential quadratic programming (SQP) algorithm, and is well-suited for large-scale, nonlinear problems. The optimizer adjusts the independent variables so that all constraints and bounds are satisfied and the final trajectory is locally optimal. With the derivatives of the objective and constraint functions with respect to the independent variables, SNOPT iteratively progresses the variables (and, thus, the trajectory) toward regions of the design space that are both feasible and locally optimal. To improve execution time and robustness, analytic derivatives are provided to SNOPT. Note that convergence to a feasible, locally optimal trajectory depends critically on the initial guess for the independent variables. Thus, an initial guess that is sufficiently close to the optimal solution is required, often a challenging task.

Several user options determine the exit conditions for SNOPT, including the "major feasibility" tolerance, the "major optimality" tolerance, the "major iteration" limit, and the "minor iteration" limit. The SNOPT output "major feasibility" indicates the degree to which the constraints and bounds are satisfied, where a lower value indicates the design is closer to feasible. Thus, when the "major feasibility" value is below the tolerance setting, the trajectory is considered feasible. Similarly, the lower the "major optimality" output value, the better the optimality conditions are met. A design is then optimal when the major optimality value is lower than the specified tolerance. For convergence, both the major feasibility and optimality values must be within tolerance.

To regulate the number of iterations that SNOPT is allowed, the parameters for the "major iteration" limit and the "minor iteration" limit parameters are specified. For every "major iteration", a quadratic programming (QP) subproblem must be solved (see Gill [73] for details), which itself requires "minor iterations". Accordingly, the minor iterations for the QP subproblem in each major iterate cannot exceed the set limit. Correspondingly, if the major iteration limit is exceeded the run will be terminated.

3. GA-GALLOP HYBRIDIZATION

The determination of a global optimum in the complex fitness landscape of low-thrust, gravity-assist trajectories requires an efficient, yet sufficiently comprehensive, exploration of the design space. It is also desirable that such a global search is automated and does not necessitate intensive user preprocessing or a priori knowledge of any optima. Recent approaches to global low-thrust trajectory optimization all possess both advantages and limitations. In this investigation, a global optimization technique for low-thrust, gravity assist trajectories is developed that overcomes several of the difficulties faced by alternative approaches. This technique hybridizes a genetic algorithm with the direct scheme in GALLOP, allowing for the exploitation of the best characteristics of the two distinct methods, while concurrently compensating for the limitations of each individual approach. The motivation and characteristics of such a combination are detailed in this chapter, along with a description of the implementation.

3.1 Hybrid Advantages

The hybridization of a genetic algorithm with a direct method such as GALLOP, produces an automated global search strategy, capable of an efficient exploration of the design space for preliminary low-thrust trajectory design. By itself, a GA is proficient in rapidly locating promising regions in an expansive, multimodal design space without the need of a user-supplied initial guess. However, the algorithm is rather inefficient in refining the solution once the general region has been identified. Additionally, the GA is typically unable to converge on the exact optimal solution due to the discretization of the design variables for the binary encoding. Moreover, when applied to the direct low-thrust, gravity assist trajectory problem, the GA struggles to meet the tight, nonlinear

constraints without exploiting gradient information. Conversely, GALLOP's formulation allows the program the ability to locally optimize complex LTGA trajectories with many intermediate bodies in an efficient and robust manner. Successful application of GALLOP does, however, require an initial guess that is suitably close to a locally optimal solution, and by itself cannot develop a global optimum.

To capitalize on the proficiencies of both of these methods, they are combined so that the GA acts as a 'wrapper' around GALLOP. As with a standard GA, the initial population is created randomly (with a uniform distribution) and the genetic operators evolve the population towards the best regions of the fitness landscape. However, in the GA-GALLOP hybrid, each member of the population is sent to GALLOP to be locally optimized. Before optimization in GALLOP, the individuals of the population are, generally, neither feasible nor locally optimal. GALLOP then refines and 'repairs' each individual design, allowing SNOPT to adjust the design variables towards feasible and locally optimal areas of the design space neighborhood. Once GALLOP has converged or reached a maximum number of iterations, the individual is assigned a fitness based on its final mass and a measure of the constraint violation (the major feasibility output from SNOPT).

3.1.1 Towards Efficiency

With a deterministic local optimization stage, the population of the genetic algorithm is improved much more precisely and efficiently than by the GA operating alone. The integration of GALLOP into the main loop of the GA allows for efficient local optimization and constraint handling because of GALLOP's incorporation of gradient information with SNOPT. The local search alleviates the GA of expending numerous generations attempting to meet the nonlinear constraints and then trying to identify optima within promising regions. In a single generation, GALLOP can efficiently converge on a feasible local optimum in an individual's design space neighborhood (if one exists), determining if the region contains a solution with a high final mass or if it is infeasible. Thus, the GA-GALLOP hybrid overcomes the GA's constraint handling and

refinement weaknesses while at the same time drastically increasing efficiency. After local optimization, the members of the population are nominally better 'adapted' to the fitness landscape topology. In this way, the Darwinian evolution analogy of the GA is extended to the hybrid formulation. The inclusion of GALLOP's local optimization is akin to an individual's ability to learn during its lifetime and become better suited for its environment; it is able to adapt.

As a gradient-based direct approach, GALLOP is particularly appropriate as the local search method in the hybrid because of the robustness inherent to its trajectory model. The large radius of convergence exhibited by GALLOP transforms the design space of the GA so that the feasible regions along with the basins of attraction of local optima are expanded as illustrated in Figure 3.1. This increase in the radius of convergence is advantageous because it decreases the complexity of the nominal GA design space, permitting identification of optima that may be relatively far from an individual's actual design variables (the starting point for GALLOP's local optimization process). Consequently, the discovery of good solutions is significantly more probable with GALLOP's direct method formulation than with a method that is more sensitive to the initial guess.

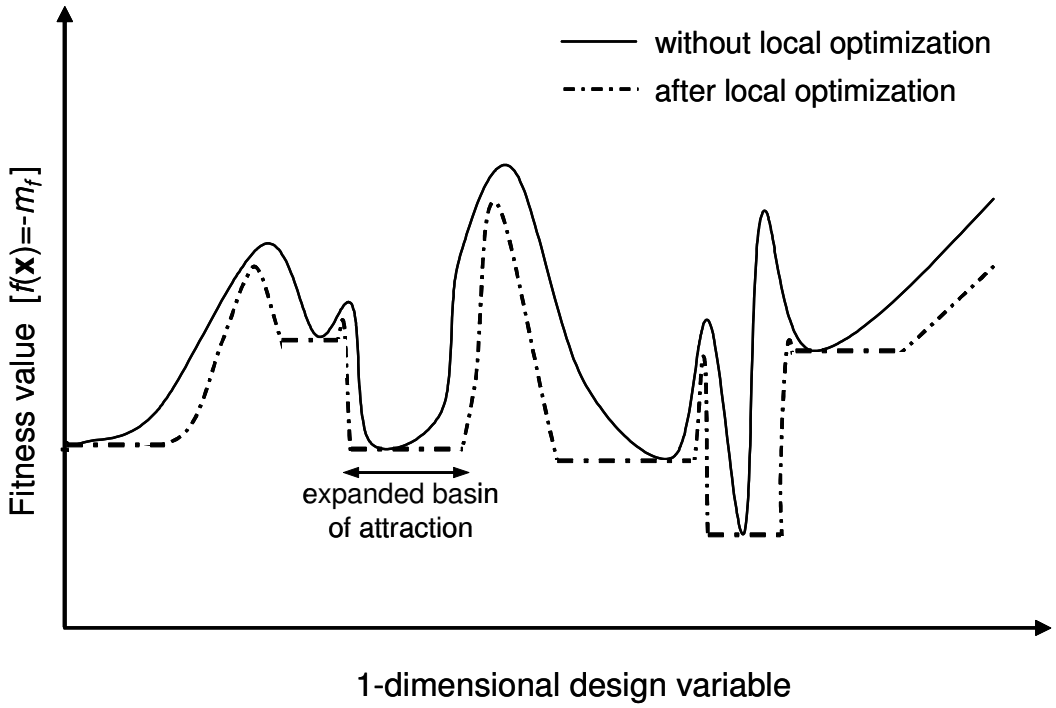

Figure 3.1 Design space modification introduced with GALLOP local optimization stage

In addition to the benefits provided by the robustness of GALLOP, the relatively short execution times demonstrated by GALLOP are fundamental to the overall efficiency of the hybrid. Because each member of the GA's population is locally optimized, a fast local routine is critical to the performance speed of the hybrid algorithm. Moreover, GALLOP has the benefit that its optimization run can be stopped after a maximum number of iterations, ensuring that the local search is reasonably short.

3.1.2 An Unbiased Global Search

Another key motivating factor for this hybridization is the elimination of a user-defined initial guess to start the optimization process. Developing a starting point is one of the most difficult aspects of trajectory optimization. This is especially true for LTGA trajectories because of the long periods of continuous thrusting in addition to the phasing issues of multiple flybys. As a calculus-based method, GALLOP is not intrinsically free of this initial guess obstacle. Although the direct method formulation of GALLOP is not

as sensitive to the initial guess as indirect methods, the optimizer still requires a sufficient basis from which to initiate the process. Furthermore, GALLOP is only capable of local optimization; however, it is an efficient and robust local scheme.

By incorporating a genetic algorithm to perform the global search, the necessity of a user developing an adequate initial guess is removed from the hybrid. The biases of any initial guess technique are also eliminated because of the random-based population initialization of a GA. Thus, no preconceptions or notions of the characteristics of optimal trajectory are built in to the search process. This lack of a bias is beneficial because it allows for the generation of nonintuitive trajectories that may be globally optimal. Moreover, additional resources are not necesssary to develop an initial guess. The GA-GALLOP hybrid algorithm executes the optimization in an automated single step without expert guidance. Thus, employing the GA as wrapper compensates for GALLOP's initial guess requirement, while simultaneously providing global exploration of the design space.

3.1.3 Hybrid Characteristics

The GA component of the hybrid provides an effective global search, but it also carries the computational expense inherent to all evolutionary algorithms due to their utilization of a population. Furthermore, the GA-GALLOP hybrid algorithm is especially costly in terms of computing resources since, in every generation, the entire population is locally optimized with GALLOP. The GALLOP simulations drive the overall computational cost of the hybrid, with the execution time ranging from a fraction of a second to several minutes. Therefore, if the hybrid is run serially, simulations for complex cases with many flybys can take longer than a week. However, the formulation of the GA-GALLOP hybrid renders the technique easily parallelizable. All of the GALLOP simulations in a generation are entirely independent of each other and can be executed simultaneously. Thus, the hybrid's run time can be dramatically reduced with parallel structure. Run times can be decreased almost linearly, and simulations on

complex cases that may require two weeks with serial processing can be shortened to a few hours.

Another distinct advantage of incorporating a genetic algorithm as the global component of the hybrid algorithm is the added capability of multiobjective optimization in a single execution of the algorithm. The population-based approach of a GA allows for the concurrent optimization of an entire set of designs. Thus, the development of a Pareto front of globally optimal solutions can be achieved by simply substituting a multiobjective genetic algorithm, such as the non-dominated sorting genetic algorithm II (NSGA-II), for the simple GA in the nominal hybrid formulation. This modified combination, the multiobjective GA-GALLOP hybrid, possesses the capability of optimizing both final mass and time-of-flight simultaneously. A mission designer can then weigh the relative importance of each objective for an informed selection.

As the local optimization element of the hybrid algorithm, the optimization efficiency exhibited by GALLOP renders it especially well suited for this hybrid technique. However, because of the simplifying assumptions built into its trajectory model, GALLOP is only a preliminary design tool. Consequently, an optimal trajectory generated by the GA-GALLOP hybrid must be ported and optimized in a higher fidelity model that includes a true, continuous thrust representation before the trajectory can be used for a mission. Furthermore, as with any global optimization method (except a prohibitively comprehensive grid search), the GA-GALLOP hybrid cannot guarantee the generation of the absolute global optimum. However, with an appropriate population size, the technique's formulation is such that the production of an optimum that is near the true global solution is highly likely.

3.2 Hybrid Design Considerations

Several design aspects of the GA-GALLOP hybrid formulation must be considered before constructing the algorithm. Previous studies on the hybridization of an evolutionary algorithm and a local search procedure aid in the development of an effective hybrid configuration. The interaction between the genetic algorithm and

GALLOP in the hybrid presents several design elements that influence the evolution of a solution, as well as the overall efficacy of the technique.

3.2.1 Memetic Algorithms

The search properties resulting from the combination of an evolutionary algorithm (EA) and a local search method are not necessarily unique to the GA-GALLOP hybrid. Since the early 1990's, several researchers have been investigating the properties of such couplings, and an entire subcategory within global stochastic techniques has been established and classified by Moscato [74] as *memetic algorithms* (MA). Typically, the term memetic algorithm includes any evolutionary algorithm that incorporates a local improvement procedure. The label 'memetic algorithms' is derived from Dawkins' concept of a "meme" as an idea transmission device, able to evolve in its cultural environment [75]. The hybridized algorithms are also denoted using various other names such as hybrid GAs, genetic local search, Lamarckian EAs, or Baldwinian EAs.

Both theoretical [76] and empirical evidence [77-80] demonstrates the effectiveness of this type of hybridization for a wide range of applications. Memetic algorithms are, generally, significantly more efficient and effective than an EA applied alone because problem-specific information or domain knowledge, can be utilized in the local search. Thus, the local search phase is specialized for a particular problem, while the EA provides a flexible, population-based exploratory search capability. That is, the local search affords an efficient exploitive mechanism to supplement the EA's general exploration process. The local search could be any type of improvement process, and is often an experience-based heuristic or a routine that exploits of gradient information. In many implementations, the local search is applied to either repair an infeasible design or provide rapid convergence to a local optimum. The local component of the GA-GALLOP hybrid accomplishes both processes simultaneously.

3.2.2 Inheritance Schemes

Extending the biological metaphor of evolutionary algorithms to memetic algorithms, the addition of the local search is analogous to the ability of individuals to learn or acclimatize to their environment during their lifetime. Genetically, this means the phenotype, the observable result of expressed genes, possesses plasticity, and can be modified with lifetime learning. For the evolutionary process to benefit from this learning (or from the ability to learn), however, these 'adaptations' must somehow affect future generations. Accordingly, after the individual is optimized in the local search, the design variables and fitness value have changed, and two genetic inheritance schemes are possible: Lamarckian or Baldwinian inheritance. The selection of the inheritance scheme strongly dictates the impact the individual's adaptation on the course of the evolution.

3.2.2.1 Lamarckian Inheritance

Lamarckian inheritance is based on the theory proposed in 1809 by the French biologist Jean Baptiste Lamarck, which claims that the acquired traits of organisms can be passed on to their progeny. That is, characteristics that are developed throughout an individual's life, not inherited, can be passed on to the next generation. Modifications in the phenotype are reflected by changes in the genotype. For example, in Lamarckian evolution, if a person has become an expert rock climber through many years of practice, their children would instinctively become great climbers. Although now replaced by Mendelian inheritance, Lamarckian evolution was widely accepted, even by Darwin, through the 19^{th} century. It was perceived as the fundamental mechanism in the generational view of a species' adaptability.

Despite a lack of evidence and general rejection in the biological community, Lamarckian evolution is widely employed in memetic algorithms. In a Lamarckian inheritance scheme, the adapted design resulting from the local search replaces an individual's original design. An inverse mapping from phenotype to genotype is necessary; that is, the genes of the individual are modified to reflect the adapted design

variables. In MAs, the modified design variables are encoded in binary to create an entirely new chromosome, or binary string, for the individual. With altered, hopefully improved, design variables from the local search, the individual now has a new objective function value and resulting fitness. The modified binary string and associated fitness are then returned to the evolutionary algorithm's current population. Thus, the adapted characteristics resulting from the local search are inheritable; the newly acquired genes are immediately able to compete within the population.

Implementing Lamarckian inheritance in the GA-GALLOP hybrid is straightforward. Before each individual is locally optimized in GALLOP, its chromosome is a binary string of encoded design variables that represent a starting point for GALLOP. The chromosome is then decoded into real-valued design variables, its phenotype, and a GALLOP input file is created. GALLOP then locally optimizes that individual, altering its phenotype, and generating a final mass and major feasibility parameter value that combine to become the individual's fitness. After optimization, the design variables associated with the solution generated by GALLOP represent the individual's altered phenotype. The new design variables are then encoded in binary to create a modified chromosome that replaces the individual's previous genotype in the population. Thus, GALLOP's local search imparts a strong and immediate impact on the evolutionary path.

3.2.2.2 Baldwinian Inheritance

The alternative inheritance scheme, Baldwinian inheritance, follows from James Mark Baldwin's theory that, although acquired traits cannot be passed on (as in Lamarckian evolution), the ability (or tendency) to develop these traits is contained in an individual's genotype and can be inherited. Proposed in 1896, this theory, now known as the Baldwin effect, is the currently supported theory for evolutionary adaptation. As in Lamarckian evolution, the phenotype of an individual can be modified to conform to an environment (phenotypic plasticity). Conversely, in Baldwinian evolution, the changes in the phenotype cannot be reflected in an altered genotype; that is, genes do not possess the capacity to change according to an environment. However, the genes do allow an

individual to adjust to its environment; and that ability can be inherited. Thus, the Baldwin effect can guide the evolution of a species towards beneficial adaptations.

In memetic algorithms, Baldwinian inheritance, like Lamarckian inheritance, uses the local search to develop an individual's fitness. However, in a Baldwinian inheritance scheme, the solution from the local search does *not* replace the original design. Thus, only the phenotype is altered, and the binary string of the individual's chromosome does not change. With Baldwinian inheritance, the improvements in the individual solution gained through the local search are not coded back into the genotype; the genes remain static throughout the lifetime of the individual, regardless of any adaptation.

The implementation of Baldwinian inheritance in the GA-GALLOP hybrid is similar to that of Lamarckian inheritance, except that the original design that is sent to GALLOP is not replaced by the locally optimized solution. The final mass and major feasibility output information from the GALLOP simulation still determine the fitness of the individual, but the chromosome remains unaltered. With Baldwinian inheritance, GALLOP still smoothes out the fitness landscape and increases the basins of attraction of the local optima. However, the influence on the evolution by the resulting designs, and the corresponding design traits, is not as powerful or as immediate as in a Lamarckian scheme.

3.2.2.3 Implications

The selection of Lamarckian or Baldwinian inheritance is not straightforward. Several empirical studies have been conducted comparing the effectiveness and efficiency of Lamarckian versus Baldwinian inheritance in memetic algorithms with conflicting results [77,80]. While the studies demonstrate that, for the particular functions examined, incorporating a local search along with a genetic algorithm is overwhelmingly more effective than using a GA alone, the conclusions differ on which inheritance scheme is superior. Thus, it is likely that the most appropriate inheritance approach is problem dependent.

The conflicting data concerning inheritance strategies is apparent in some notable studies. Whitley et al. [80] concluded that Lamarckian and Baldwinian strategies performed equally well in the first of three test functions. However, given the second test function, a Baldwinian approach was more likely to generate the global optimum than a Lamarckian approach. This result was offset by the Lamarckian approach succeeding more often than the Baldwinian approach in the third and most difficult test function. In all the test functions, it was clear, though, that Lamarckian inheritance was the more efficient scheme, demonstrating much faster convergence. However, the authors emphasize that a Baldwinian approach maintains better overall diversity, mitigating premature convergence. A slightly more complex examination of inheritance is reflected in a study by Houck et al. [77]. A specified percentage of the population employed Lamarckian evolution, while the rest was subjected to Baldwinian inheritance. It was discovered that using some Lamarckian inheritance in the population made convergence on the global optimum more likely and required fewer generations than employing only Baldwinian inheritance on the entire population.

Most recent implementations of MAs use purely Lamarckian inheritance or some combination of the two schemes [81]. For the GA-GALLOP hybrid algorithm, incorporating some Lamarckian inheritance, and allowing the optimized solution from the GALLOP to immediately impact the evolutionary progression should improve efficiency. However, it may be detrimental to allow infeasible solutions from GALLOP to strongly influence the evolution. The inheritance of traits corresponding to infeasible trajectories from GALLOP with a Lamarckian approach may emphasize deficient areas in the population. Furthermore, it is also essential that local optima with large basins of attraction do not quickly overrun the population, and drive convergence to a local optimum. Thus, Baldwinian inheritance is included as an option to encourage diversity in the population.

3.2.3 Genetic Operators

Selection of the genetic operators for a hybrid genetic algorithm is critical to performance. Appropriate versions of selection, crossover, and mutation operators are required to ensure proper exploration and population diversity preservation. Maintaining a diverse population is important for memetic algorithms because they often suffer from premature convergence, particularly when a high percentage of the population employs Lamarckian inheritance. Thus, applying uniform crossover over single-point crossover is advantageous for MAs with any Lamarckian evolution because of the increased exploratory capacity. Furthermore, a relatively high mutation rate may be necessary to encourage diversity. For complex problems, both single objective and multiobjecitve, elitism is highly beneficial, so that previously discovered high-performing solutions are not lost in subsequent generations.

3.3 Implementation of the GA-GALLOP Hybrid Algorithm

The GA-GALLOP hybrid algorithm is developed to provide an efficient global optimization technique for low-thrust, gravity-assist trajectories. The hybridization is designed so that the genetic algorithm and GALLOP work synergistically to maximize the advantages of each method, while minimizing the drawbacks. The hybrid algorithm is designed for both single objective and multiobjective implementations.

3.3.1 Single Objective Hybrid Structure

In the single objective implementation of the GA-GALLOP hybrid, the only objective is the maximization of final mass (minimization of $-m_f$); and, thus, the search is designed to locate a single global optimum. The structure of the GA-GALLOP hybrid is such that a simple genetic algorithm 'drives' GALLOP. As the local search engine, GALLOP is placed in the main loop of the genetic algorithm so that for every generation, each

trajectory in the population is locally optimized. A flowchart depicting the configuration of the GA-GALLOP hybrid algorithm is illustrated in Figure 3.2. The first step in the algorithm is the generation of a random initial population of trajectories (uniform distribution). This random generation is a key advantage of utilizing a GA; no preconceptions or design experience is required to initiate the optimization process. The difficulties associated with attempts to generate an initial guess for low-thrust trajectory optimization are bypassed via an automated, unbiased routine. Next, each trajectory in the population is sent to GALLOP; this step is also the start of the main loop of the GA-GALLOP hybrid. GALLOP attempts to adjust each individual in the population to deliver a feasible and locally optimal solution, overcoming the constraint and numerical accuracy limitations of the GA. The GA does not manage any trajectory constraints directly (although, additional constraints could easily be incorporated into a penalty function without modifying GALLOP). Each locally optimized trajectory from GALLOP is then assigned a fitness value based on its final mass as well as the "major feasibility" output from SNOPT. The fitness function is a combination of the objective function and a penalty function, where

$$\varphi(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x}) \tag{3.1}$$

such that

$$f(\mathbf{x}) = -m_f \tag{3.2}$$

and

$$P(\mathbf{x}) = \begin{cases} 0 & \text{if } major\ feasiblity < \text{tolerance} \\ R * (major\ feasiblity) & \text{if } major\ feasiblity \geq \text{tolerance} \end{cases}. \tag{3.3}$$

The parameter R is the user-defined penalty parameter, typically set to a very large value to heavily penalize infeasible designs. Note that, if GALLOP is able to modify the trajectory so that it is feasible, that particular individual is not penalized, and possesses a higher probability of passing on its characteristics.

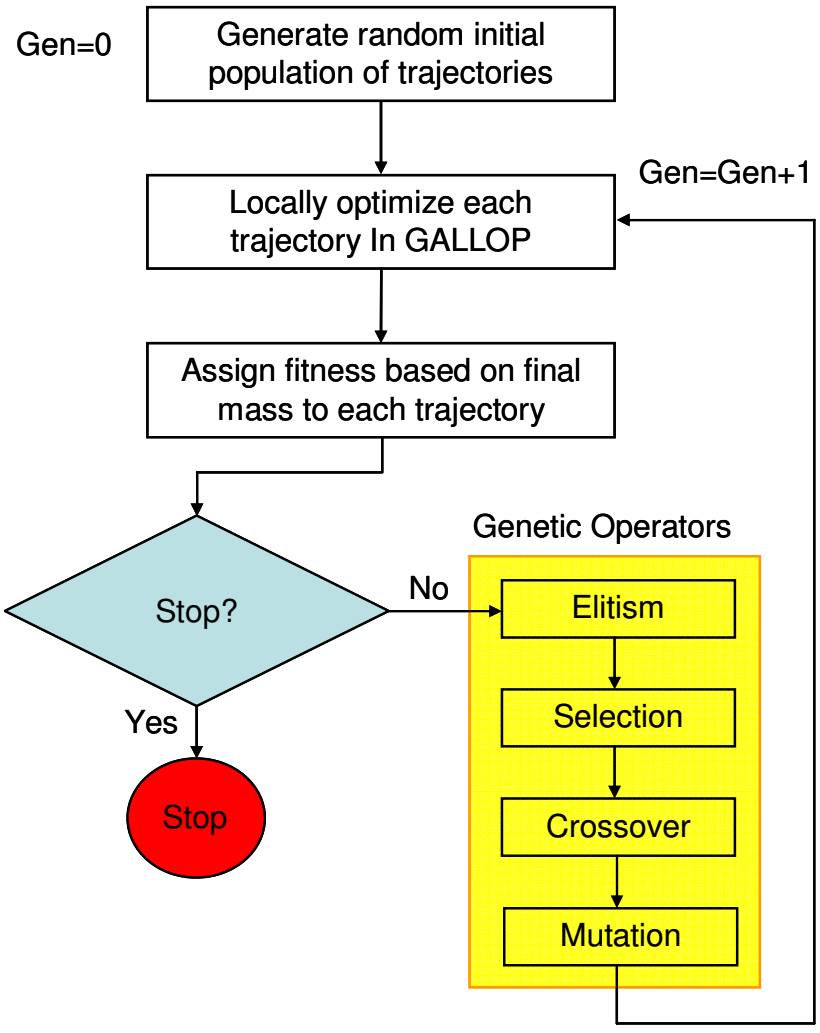

Figure 3.2 Single objective GA-GALLOP hybrid structure

Once the design variables are returned from GALLOP, the selected inheritance scheme is invoked. The hybrid allows for a combination of Lamarckian and Baldwinian inheritance, where a user-defined probability value determines which scheme is applied for each individual. Thus, several inheritance options are available for the population: 100% Lamarckian, 100% Baldwinian, or a specified ratio of the two approaches. Additional control is afforded by a 'buffer' parameter defining a maximum major feasibility value, below which Lamarckian inheritance can be employed for an individual. This buffer allows the user the ability to apply Lamarckian inheritance only to those solutions that are feasible or nearly feasible. A trajectory that is close but not quite feasible is indicated by a low major feasibility value (yet above the major feasibility tolerance), implying that the trajectory is nearly meeting all constraints. All individuals not using a Lamarckian inheritance scheme employ Baldwinian inheritance. The application of Lamarckian inheritance to only feasible (or nearly feasible) solutions increases the influence of the trajectory characteristics from feasible solutions on the evolution, while moderating the impact of trajectory characteristics from solutions that are not successful in terms of feasibility. Lamarckian inheritance applied with infeasible solutions could be detrimental to the evolutionary process since GALLOP's local optimization process may subsequently direct an individual to the infeasible regions of the design space, decreasing the overall efficiency and performance of the hybrid.

Once each member of the population has been locally optimized and assigned a fitness value, two stopping criteria are checked. In this implementation, the optimization is halted if the maximum number of generations has been reached or if there is only a slight change in the average fitness value for several consecutive generations, indicating convergence. If a stopping criterion is not met, the entire population of solutions is sent to the core of the GA, the genetic operators.

The global search of the design space is implemented through the genetic operators, mixing exploration and exploitation to evolve the population. Unless it is the first time through the main loop of the hybrid (i.e., the first generation), elitism is applied to ensure that the best trajectories from the previous generation are not lost. The elitism operator retains a specified percentage of the top trajectories from the previous generation and

replaces the trajectories with the worst fitness in the current population (keeping the population size the same). The selection operator models a "survival of the fittest" competition to filter out the best trajectories from the current population. In this implementation, tournament selection is employed to ensure that the best trajectory in the population mates twice. These 'better-adapted' designs are then directed to the crossover operator that combines the characteristics, or the genes of the designs, to generate an entirely new population of trajectories. Because memetic algorithms are sometimes susceptible to premature convergence (especially with Lamarckian inheritance), uniform crossover is incorporated. Next, to further explore the design space, the new offspring population is mutated. A clock mutation operator introduces random genetic patterns to create trajectory characteristics that may not currently exist in the population. The loop is then restarted by returning the newly created population back to GALLOP to be refined. Through this hybrid formulation, the strengths of each method are used to compensate for the shortcomings intrinsic to either method alone. A search for globally optimal solutions is accomplished, unhindered by initial guess requirements.

Several parameters must be specified prior to initiating the optimization procedure with the GA-GALLOP hybrid algorithm, including the launch vehicle model, engine model, and solar array model. Additionally, the GA requires specification of the population size, the maximum number of generations, the mutation probability, and the elitism percentage. The population size is especially influential in the success of the hybrid algorithm. For large problems, in which wide ranges of encounter dates are examined, a sufficiently large population is necessary to adequately explore the design space. The mutation probability and elitism percentage are also important in the progress of the evolution, affecting the exploration/exploitation balance of the genetic operators. For GALLOP and SNOPT, appropriate values for the major feasibility parameter and optimality tolerance, as well as major and minor iteration limits are required. Both the tolerances and the iteration limits should be specified to allow GALLOP an opportunity to converge to the local optimum while at the same time ensuring that the simulations are as brief as possible. Lastly, two parameters must be defined for the inheritance strategy associated with the design variables that are output from GALLOP: the Lamarckian

probability and the Lamarckian buffer. The Lamarckian probability value defines the percentage of the individuals of the population that employ Lamarckian inheritance. The Lamarckian buffer is the maximum major feasibility value that a solution from GALLOP may possess and still use Lamarckian inheritance; all other individuals use Baldwinian inheritance.

3.3.2 Multiobjective Hybrid Structure

The multiobjective implementation of the hybrid is similar in structure to the single objective version. The fundamental difference is the dual objective nature, that is, in the multiobjective hybrid algorithm, both the maximization of final mass and the minimization of the time-of-flight are optimization objectives. Thus, instead of employing a simple genetic algorithm, the Non-dominated Sorting Genetic Algorithm II is utilized to accomplish the genetic search. The formulation of the multiobjective version is demonstrated in Figure 3.3.

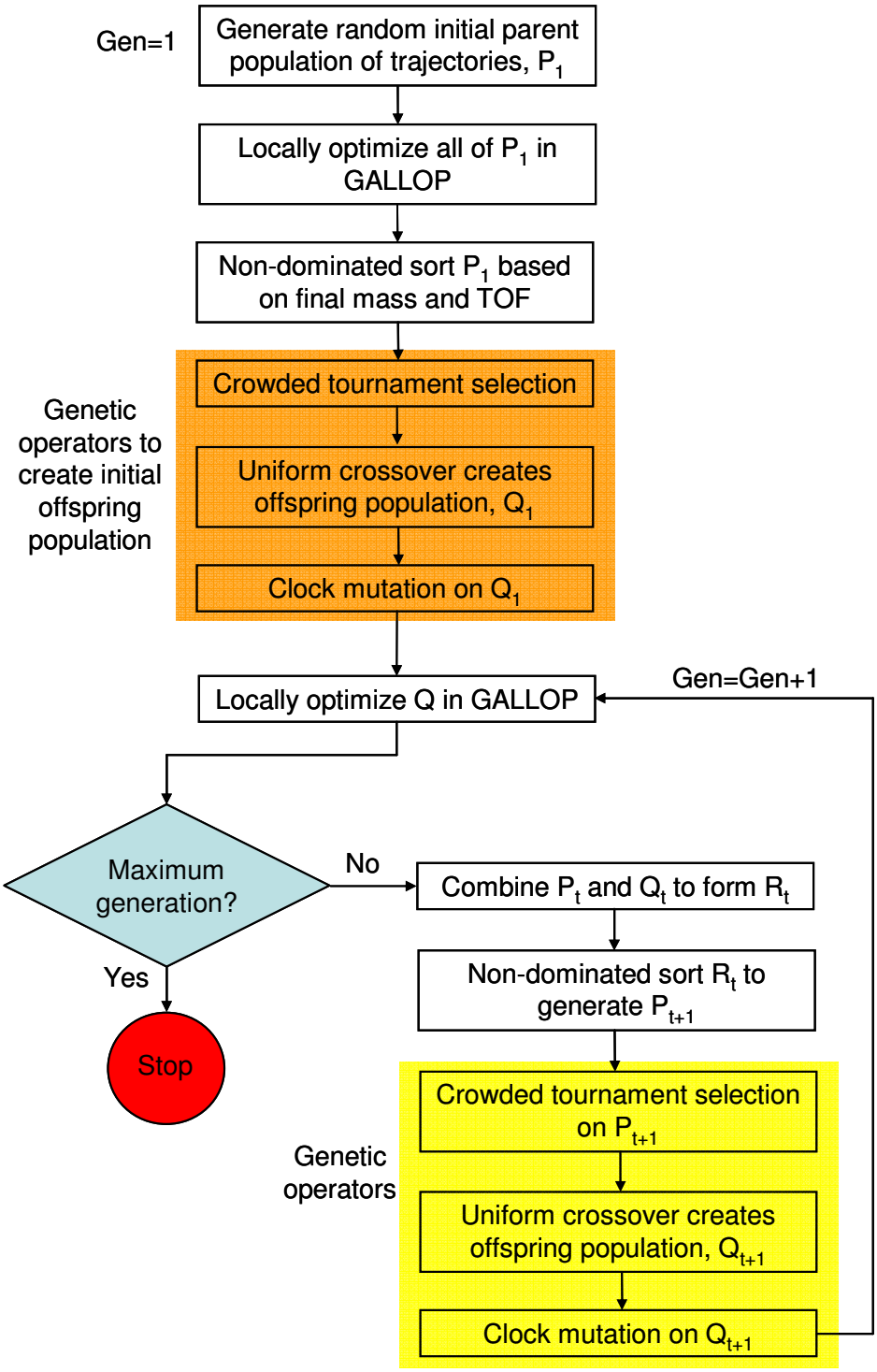

Figure 3.3 Multiobjective GA-GALLOP hybrid structure

The initial step in the multiobjective algorithm is similar to the single objective implementation. The initial parent population, P_1, of size N is created with a uniform random number generator. However, in the multiobjective version, there are two objectives and the fitness value is developed using a non-dominated sorting routine. An attempt to locally optimize P_1 is accomplished in GALLOP; then, a user-defined penalty value is added to both the final mass and the time-of-flight if the individual solution is not feasible. The penalized objectives are employed in the sorting routine to develop the various non-dominated fronts. The fitness of an individual is assigned according to the rank of the non-dominated front to which it belongs. Recall that the closer the front is to the Pareto front, the lower the fitness value of individual. At this stage, the inheritance of the design variables from GALLOP are also determined. The inheritance options are the same in the multiobjective hybrid as in the single objective hybrid algorithm. Thus, each individual in the population can use Lamarckian or Baldwinian inheritance based on user-defined parameters.

A significant difference between the multiobjective and single objective versions of the hybrid is the development of an offspring population of trajectories in the multiobjective GA-GALLOP hybrid. The initial offspring population, Q_1, of size N is generated by application of the genetic operators: crowded tournament selection, uniform selection, and clock mutation. Once Q_1 is created, the procedure enters the main loop of the hybrid algorithm, and each member in the offspring population is locally optimized with GALLOP. Consistent with the initial parent population, the inheritance in the offspring population is determined via user-defined parameters. After the final mass and the time-of-flight corresponding to the infeasible solutions are penalized accordingly, the stopping criterion is checked. If the number of generations is greater than a specified maximum, the process is terminated. Otherwise, the offspring solution is then combined with the parent population, creating a combined population, R_t, of size $2N$. The use of both a parent and offspring population allows for the incorporation of elitism by maintaining the elite individuals from a generation in a new parent population.

In the main loop of the hybrid algorithm, the combined population is ranked with a non-dominated sorting. The new parent population, P_{t+1}, is generated from the

individuals in the best non-dominated fronts, where the last available slots in the new population are filled based on the crowding distance. Thus, this parent population is composed of the elites of the combined population and generates the next offspring population. The new offspring population, Q_{t+1}, is developed in the same way as the initial offspring through crowded tournament selection, uniform crossover, and clock mutation. The loop then restarts, with each individual in the new offspring population passing to GALLOP. Except for the elitism percentage, the same user parameters as in the single objective implementation are required for the multiobjective hybrid.

3.3.3 Design Variables

The design variables for the GA-GALLOP hybrid are the independent variables from the GALLOP formulation (see Table 2.4). These include the launch and encounter dates, the spacecraft mass at each body, the incoming and outgoing relative velocity vectors, the B-plane angle, the flyby radius, and the three components of the thrust vector at the midpoint of every segment. These design variables are the same in both the single objective and multiobjective versions of the hybrid. As with the standard GA, each variable is coded in binary and concatenated to form the chromosome of an individual. The numerous thrust vectors on each leg comprise the majority of GALLOP's independent variables, driving the length of the binary string representing the chromosome. The longer the binary string, the more generations are required for the GA to develop the optimal patterns of 1's and 0's. Thus, reducing the number of parameters required to effectively represent the thrust can reduce convergence time.

No initial guess for any of the design variables is necessary. All that is required is the definition of a lower and upper bound, as well as the number of bits desired to represent each design variable. If the lower and upper bounds are set to the same value, the variable is removed from the search and is not represented in the chromosome. The ranges of the lower and upper bounds for the launch date and encounter dates truly drive the size and complexity of the design space. The number of local optima increases sharply with the widening of the date ranges. Thus, the necessary population size is

correlated with the difference between the lower and upper bounds for the dates; when the range is wide, the population size must also be large to increase the likelihood of discovering the global optimum. In general, except for the minimum flyby radius, the bounds on the other independent variables (relative velocity vector components, mass, etc.) should be free to ensure a global search within the specified date ranges; that is, the bounds should be set reasonably wide to encompass all physically feasible trajectories within these dates. However, the larger the specified range, the more bits are required to maintain a desired resolution within the genetic algorithm.

3.3.4 Reduced Thrust Parameterization

To reduce the number of design variables that are required to represent the thrust vectors in the genetic algorithm, a series approximation is used for the thrust angles along each segment in GALLOP. Within GALLOP, the thrust vector is represented in terms of spherical coordinates, that is, the magnitude of the thrust, a clock angle, θ, and a cone angle, ψ. In the GA, external from GALLOP, the two thrust angles are modeled as functions of the normalized leg duration. The two angle functions are then approximated with a Chebyshev series [82], using a linear least squares fit. Modeling the two thrust angles with a Chebyshev series to reduce the number of parameters for a direct thrust representation was successfully implemented previously by Yam and Longuski [83]. For example, the clock angle is represented as:

$$\theta(u, c_1, .., c_k) = \sum_{k=0}^{N} c_k T_k(u) \tag{3.4}$$

where N is the degree of the Chebyshev series, c_k are the coefficients of the series, $T_k(u)$ is a degree k Chebyshev polynomial of the first kind, and u is the normalized leg duration spanning from 0 to 1. With this approximation, the coefficients of the Chebyshev series become the independent design variables for the GA, replacing the two thrust components on every segment along the leg. Thus, the degree of the Chebyshev series is specified along with lower and upper bounds for each of the N+1 coefficients on every

leg. This approximation can significantly reduce the number of design variables representing the two thrust angles. For example, on a trajectory with four legs, each with 25 segments, if the user specifies a Chebyshev series of degree 5 for both angle representations, the GA handles 48 design variables using the Chebyshev approximation compared to 200 variables without the approximation. However, one drawback of this approach is the fact that the Chebyshev series may not always be a sufficiently accurate representation of the thrust profile, and trajectory characteristics that are beneficial to the evolution can be lost with the approximation. Additionally, an appropriate degree and suitable bounds for the Chebyshev series must be determined.

A further reduction in the number of variables is accomplished by assuming an on/off formulation for the thrust magnitude on each segment. Such an approximation is appropriate because, on most optimal low-thrust trajectories, the thrust magnitude is either at its maximum or zero (coasting). This assumption does not reduce the number of design variables, but does decrease the number of binary bits necessary to encode the thrust magnitude, minimizing the chromosome length. A "1" indicates maximum thrust on the segment, whereas a "0" designates a coasting segment. This on/off formulation does not require the specification of lower and upper bounds or the number of bits for the thrust magnitude variables. In all test cases, both the Chebyshev series and on/off approximations are applied. Note that, in addition to defining the bounds and number of bits for the coefficients in the Chebyshev series, the number of segments for each leg must also be specified.

.

4. RESULTS

To determine the efficacy of the GA-GALLOP hybrid formulation, both the single objective and multiobjective implementations are applied to challenging low-thrust trajectory design scenarios, and the results are compared to those in the literature.

4.1 Single Objective Hybrid Results

The single objective implementation of the GA-GALLOP hybrid algorithm is tested on four different low-thrust trajectory design examples with the goal of generating the global optimum in terms of final mass. The four examples include: 1) a direct rendezvous trajectory from Earth to Mars using solar electric propulsion; 2) an Earth-Venus-Earth-Jupiter-Pluto rendezvous trajectory using nuclear electric propulsion; 3) an Earth-Mars-Earth-Jupiter-Pluto rendezvous using NEP; and 4) an Earth-Venus-Earth-Jupiter-Neptune rendezvous using NEP. Note that the flyby sequence in the examples was predetermined and was not optimized.

4.1.1 SEP Earth-Mars Rendezvous

For verification of the hybrid algorithm, the optimization of a relatively simple solar electric propulsion low-thrust trajectory is first considered. To demonstrate the ability of the GA-GALLOP hybrid to generate a known global optimum, the technique is applied to an Earth-Mars (EM) rendezvous example previously studied by Williams and Coverstone-Carroll [84]. Williams and Coverstone-Carroll conducted a parametric study

using the software SEPTOP [36] to generate parametric curves for EM trajectories with fixed flight times from 1.5 to 3.5 years and launch dates from mid-2004 to late 2009. For comparison, the same launch date range along with the same spacecraft parameters (Table 4.1) are then used in the GA-GALLOP hybrid. The Delta II launch vehicle capability is slightly lower from that in Ref. 84, delivering 1293 kg instead of 1300 kg to a launch v_∞ of zero. (See Table 4.2 for values) An NSTAR engine is used with the mass flow rate and thrust as polynomial functions of the power processing unit (PPU) input, P_a, in kilowatts. As conservative measures, 400 W are reserved for spacecraft (s/c) system operations other than thrust, and a duty cycle reduces the mass flow rate and thrust from their nominal value. The hybrid algorithm is applied with the parameters and date ranges represented in Tables 4.3 and 4.4 for a maximum time-of-flight of 1.5 years. All other design variables are "free" to ensure a global search (e.g., the bounds on the arrival mass are 0 and 1293 kg). With a population size of 100, Lamarckian inheritance is employed if the major feasibility value returned from SNOPT is less than 1.5E-4, otherwise Baldwinian evolution is used. For this example, there are 40 design variables with a corresponding chromosome length of 190 bits for each individual.

Table 4.1 SEP spacecraft parameters

Parameter	Value
Mass flow rate [mg/s]	$0.74343 + 0.20951P_a + 0.25205P_a^2$ (NSTAR)
Thrust [mN]	$3.4318 + 37.365P_a$
Duty cycle	90%
P_0	5 kW
Min/Max power to PPU	0.649 kW / 2.53 kW
$\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5$	1.1063, 0.1495, -0.299, -0.0432, 0
s/c power allocation	400 W
$\beta_1, \beta_2, \beta_3, \beta_4$	1, 0, 0, 0

Table 4.2 Launch vehicle parameters (Delta 7925, 5% contingency)

Parameter	Value
ξ_1	5582.9 kg
ξ_2	139.1 kg
ξ_3	2.3728 km/s
ξ_4	0.0 kg
Parking orbit altitude	185 km
Launch contingency	5%
Adapter mass	0 kg

Table 4.3 GA-GALLOP hybrid parameters for Earth-Mars rendezvous example

Parameter	Value
Population size	100
Elitism percentage	5%
Mutation probability	3%
Lamarckian probability	100%
Lamarckian buffer	1.5E-4
Maximum generation	100
Degree of Chebyshev series	4
Penalty parameter	1E5
Major iterations limit	300
Major feasibility tolerance	1E-4
Major optimality tolerance	1E-4

Table 4.4 GA-GALLOP hybrid variable ranges for Earth-Mars rendezvous example

Design Variable	Lower Bound	Upper Bound
Launch date	1 May 2004	31 Dec. 2009
Launch v_∞	0 km/s	10 km/s
Mars arrival date	1 May 2004	29 Apr. 2014
Mars arrival v_∞	0 km/s	0 km/s
Mars arrival mass	200 kg	1300 kg

The evolution of the variables for launch date and arrival date through the generations is reflected in Figs. 4.1 and 4.2. The respective date variable corresponding to every design in the population is plotted versus its generation number. Dark blue indicates that the design returned from GALLOP is feasible, whereas dark red identifies designs with very high major feasibility values. It is clear that by the 15th generation, the algorithm has converged on a distinct band of launch and arrival dates. The best trajectory is developed in the 6th generation as indicated in Fig. 4.3. Recall that problem is formulated as a minimization of $-m_f$ (negative of final mass); thus, the lower the fitness, or the more negative, the better the trajectory. The average fitness of the population through 50 generations is illustrated in Fig. 4.4, where the fitness initially increases sharply and then decreases before steadying near the best fitness value after generation 10. The percentage of designs that GALLOP returns as feasible for each generation is plotted in Fig. 4.5. The percent feasible rises quickly in the first few generations, then steadily increases through the 25th generation before fluctuating between 80% and 90%. It is clear from Figs. 4.1-4.5 that the most dramatic changes in the population's evolution occur in the earlier generations; and by the 15th generation the population mostly consists of high performing, feasible designs. After this initial exploration period, the search is narrowed and focused on a tighter range over each of the design variables. This tighter focus is on the values of the design variables represented in the high-performing trajectories of the later generations. The hybrid algorithm does not converge on a single trajectory (by the 50th generation) because there are several local minima with fitness values very similar to that of the apparent global optimum.

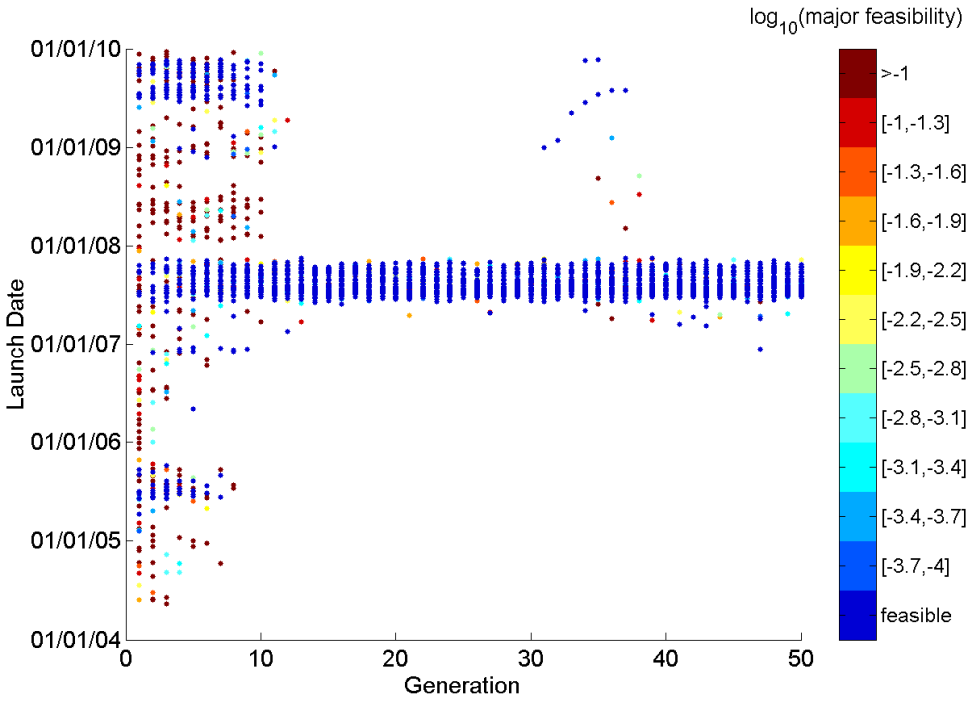

Figure 4.1 Earth launch date versus generation number for EM example.

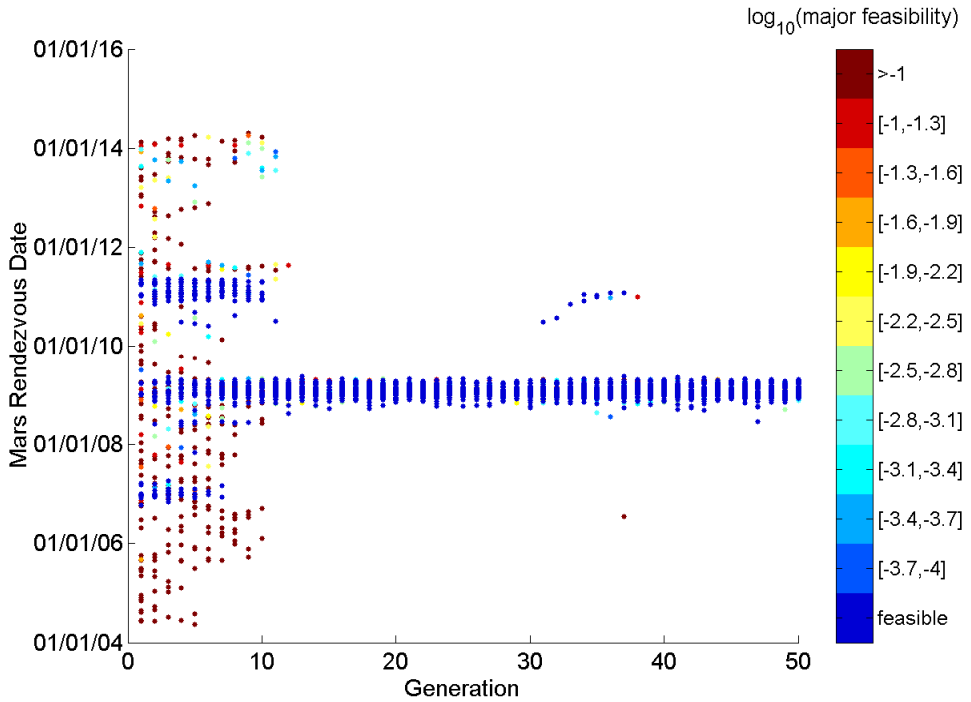

Figure 4.2 Mars arrival date versus generation number for EM example.

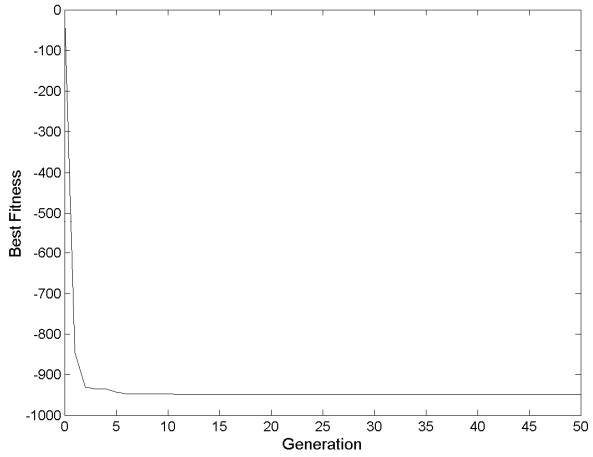

Figure 4.3 Fitness of best design through 50 generations for EM example.

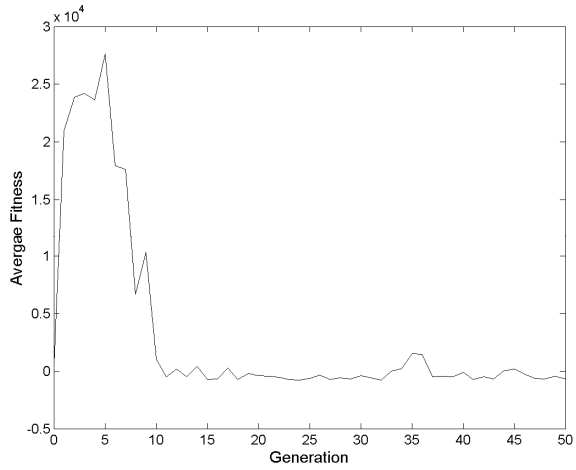

Figure 4.4 Average fitness of population through evolution of EM example.

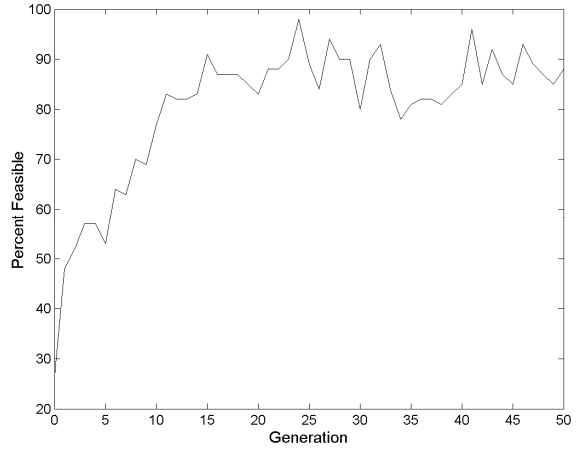

Figure 4.5 Percentage of feasible designs through evolution of EM example.

After 50 generations, the best 1.5-year trajectory from the GA-GALLOP hybrid algorithm is "re-optimized" in GALLOP with tighter "major feasibility" and "major optimality" tolerance values (1E-6 for both) to refine the solution. This refinement step only slightly shifted the design variables associated with the solution and did not alter the final mass (i.e., the solution corresponds to the same local optimum). The best trajectory from the hybrid algorithm possesses a final mass of 943 kg, and a launch date on August 26, 2007, consistent with the best 1.5-year trajectory from the Williams and Coverstone-Carroll analysis. This comparison is demonstrated in Fig. 4.6, where the evolution of the launch date for 40 generations from the hybrid algorithm is superimposed on the plot of the final mass versus launch date from Ref. 84. It is clear that the three distinct bands of trajectories with feasible launch dates (dark blue) from the GA-GALLOP hybrid are aligned with the 1.5-year TOF contours from Ref. 84 (contour peaks are circled in orange). The earliest band, with launch dates around June 2005 corresponds to the 1.5-year contour with the lowest final mass peak, and does not 'survive' as far into the evolution as the other two regions. The middle band, with launch dates between June 2007 and December 2007, reflects the best performing contour, and eventually out-competes the other launch opportunities. The launch date associated with the best trajectory corresponds to the launch date at the peak of the best 1.5-year contour from Ref. 84. The resulting optimal trajectory is plotted in Fig. 4.7, and its characteristics are listed in Table 4.5. Note that the trajectory is continually thrusting, where the red lines identify the thrust direction (away from the black dot, which is point on the trajectory of the spacecraft).

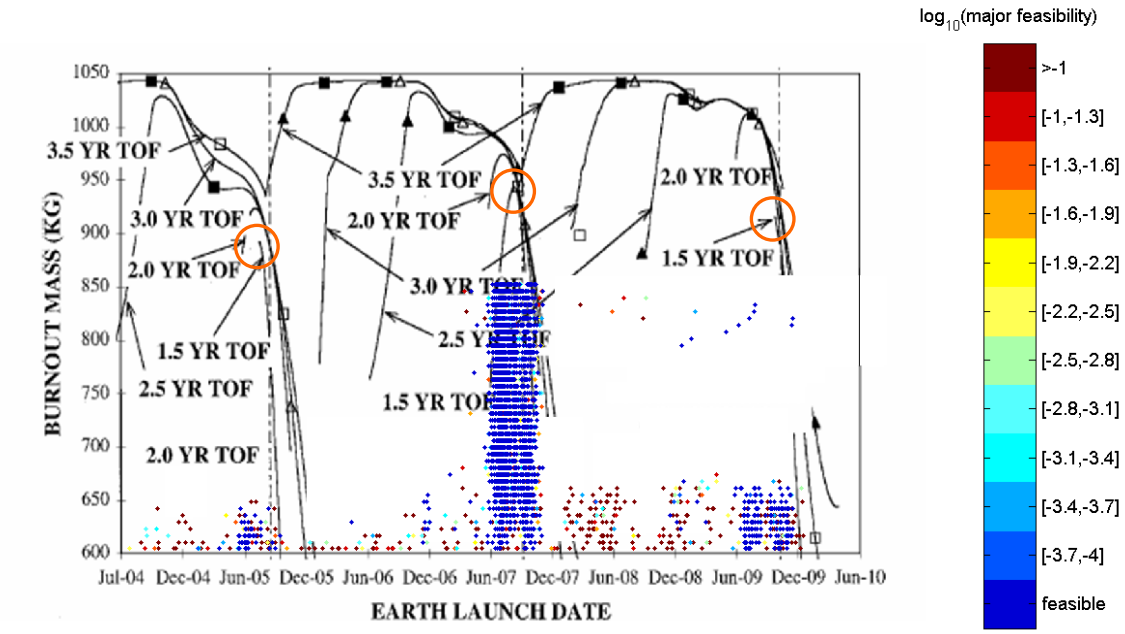

Figure 4.6 Evolution of Earth launch date for Earth-Mars rendezvous example (TOF ≤ 1.5 years) (from Williams and Coverstone-Carroll [84] with permission.)

Table 4.5 GA-GALLOP hybrid parameters for Earth-Mars rendezvous example

Characteristics	Value
Earth launch date	26 Aug. 2007
Earth launch v_∞	2.75 km/s
Injected mass	1048 kg
Mars arrival date	24 Feb. 2009
Final mass	943 kg
TOF	547.5 days

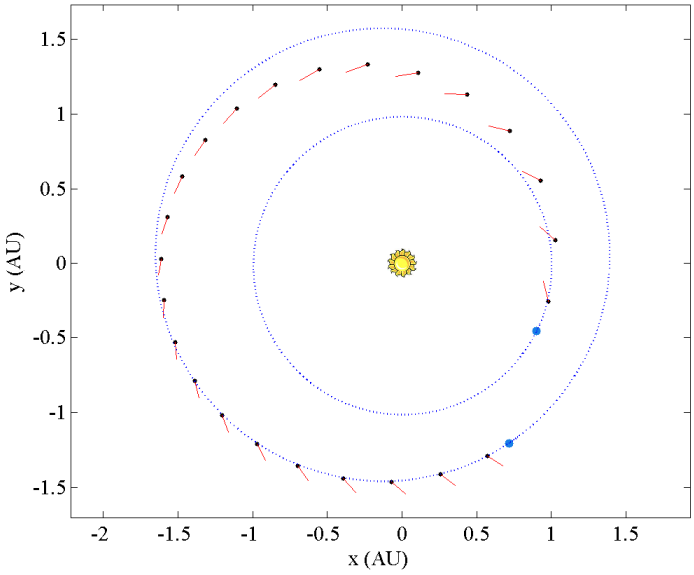

Figure 4.7 Ecliptic projection of the optimal Earth-Mars rendezvous trajectory (TOF≤1.5 years).

4.1.2 NEP Earth-Venus-Earth-Jupiter-Pluto Rendezvous

With the successful generation of the apparent global optimum for a relatively simple EM case, without any gravity assists, the GA-GALLOP hybrid algorithm is applied to a much larger-scale problem to validate its proposed efficacy. That is, the hybrid algorithm is tested on a complex case with several intermediate flybys over a wide range of encounter dates. For comparison, the comprehensive study of nuclear electric propulsion missions to the outer planets by Yam et al. [85,86] is fitting. Yam et al. use a two-step process to optimize direct and gravity-assist trajectories to Jupiter, Saturn, Uranus, Neptune, and Pluto. The authors first conduct a broad search of the design space with STOUR-LTGA, then select the best candidates as initial guesses for local optimization with GALLOP. This type of approach was similarly employed to produce the winning trajectory in the first Global Trajectory Optimisation Competition in 2005 [87], and effectively generated many high performance trajectories in Yam et al. [85]. Thus, the

studies in Yam et al. [85,86] are excellent cases for comparison with the GA-GALLOP hybrid.

The first example is an Earth-Venus-Earth-Jupiter-Pluto (EVEJP) rendezvous trajectory from Yam et al. [85]. Yam et al. note a level of difficulty in the determination of suitable initial guesses for trajectories to Pluto. The high-energy trajectories required to reach Pluto may not always be well represented by the exponential sinusoid shape (tangential thrust) employed in STOUR-LTGA. Thus, the GA-GALLOP hybrid approach should benefit from the lack of a requirement for an initial guess. The spacecraft parameters from Ref. 85, listed in Table 4.6, are repeated in the hybrid algorithm test run. Note that the launch vehicle is a hypothetical "heavy lifter" from Yam et al. [86]. The parameters for the GA-GALLOP hybrid are summarized in Table 4.7, and the ranges for the key design variables are listed in Table 4.8. Possible launch dates for a hybrid design range from 2010 to 2022, which varies slightly from the 2014 to 2025 range in Ref. 85. The ranges corresponding to launch and encounter dates are fairly broad, a 12-year range for the launch date and a 19-year range for the Pluto rendezvous date. Broad ranges require a higher population size (200) than in the EM test case. The time-of-flight of the trajectory is constrained to be less than 12 years so that a comparison with Ref. 85 is available. Note that an individual design from GALLOP will incorporate Lamarckian inheritance if its major feasibility value is less than 1E-3, where 1E-4 indicates a feasible design. Elitism is again employed so that the top performing 7% of the population replaces the bottom 7% of the subsequent generation. This relatively high percentage implies that the characteristics of the best trajectories exert a large influence on the course of the evolution. There are 171 design variables corresponding to 983 bits per chromosome for this case, a significant increase from the EM example.

Table 4.6 NEP spacecraft parameters

Parameter	Value
Launch vehicle	"heavy lifter" (20,000 kg to v_∞=0)
Power available to thrusters	95 kW
Specific impulse	6000 s
Overall efficiency	70%
Thrust	2.26 N
Mass flow rate	38.4 mg/s
Duty cycle	100%

Table 4.7 GA-GALLOP hybrid parameters for EVEJP rendezvous case

Parameter	Value
Population size	200
Elitism percentage	7%
Mutation probability	3%
Lamarckian probability	100%
Lamarckian buffer	1E-3
Maximum generation	150
Penalty parameter	1E5
Major iterations limit	1000
Major feasibility tolerance	1E-4
Major optimality tolerance	1E-4
Degree of Chebyshev series	5

Table 4.8 GA-GALLOP hybrid key design variable bounds for EVEJP rendezvous case

Design Variable	Lower Bound	Upper Bound
Launch date	1 Jan. 2010	1 Jan. 2022
Launch v_∞ [km/s]	0	10
Venus arrival date	1 Feb. 2010	1 Jan. 2030
Venus flyby altitude [km]	500	10,000
Earth arrival date	1 Mar. 2010	1 Jan. 2032
Earth flyby altitude [km]	500	10,000
Jupiter arrival date	1 Jan. 2011	1 Jan. 2033
Jupiter flyby altitude [R_J][a]	2	40
Pluto arrival date	1 Jan. 2015	1 Jan. 2034
Final mass [kg]	2,000	20,000

The evolution of the launch and encounter dates, along with the launch v_∞, through 150 generations, is plotted in Fig. 4.8. Initially, the entire design space is broadly surveyed, indicated by the uniform spread of potential launch and encounter dates in the first few generations. However, by the 25th generation, distinct bands of feasibility have formed on each of the plots for launch and encounter dates. Subsequently, most of the exploration is conducted within these date ranges. This early narrowing of the feasibility bands demonstrates the capability of the GA-GALLOP hybrid algorithm to distinguish promising regions of the design space efficiently, so that these regions become the focus of the search. In Fig. 4.8b, trajectories with low relative launch velocities (below 4 km/s) are originally established as being generally infeasible, and thus are not promoted extensively. However, after generation ~20, feasible high-performing trajectories with low relative launch velocities are progressively discovered, reemphasizing low launch v_∞ within the search process.

By the 150th generation (30,000 GALLOP evaluations), it is apparent that there are several competing launch opportunities. Furthermore, narrow bands of Venus and Earth flyby dates have emerged, indicating two optimal phasing occurrences between the two

planets. The feasible bands for the Jupiter flyby and the Pluto rendezvous date are distinct but rather wide, where the blue band for both planets spans over two years. If the GA-GALLOP hybrid is continued for more generations, the algorithm should eventually converge on a single design. The fitness corresponding to the best design in each generation is plotted in Fig. 4.9, illustrating that the most dramatic changes in the evolution of the best fitness value in the population occur in the earlier generations (before generation ~50). This behavior is corroborated in Fig. 4.10, where the average fitness of the population is plotted versus generation number (recall that it is a minimization problem). The average fitness decreases quickly during the first 20 generations and then steadies, fluctuating between 0 and 3E5 through the rest of the evolution. Correspondingly, the percentage of feasible solutions increases sharply through the initial 50 generations, reaching a peak value equal to 25% feasible and, then, fluctuating through the the 150th generation. It is expected that the hybrid algorithm would eventually converge on a single trajectory; however, a maximum generation number equal to 150 is considered sufficient based on the progression of the best fitness value.

The hybrid algorithm generated several high-performing trajectories, most with a TOF near the 12-year upper limit. The best solution, with a final mass of 10,632 kg and a time-of-flight of 11.99 years, is generated in the 75th generation. (As in the EM example, the solution was re-optimized with tighter "major feasibility" and "major optimality" tolerances, 1E-6 for both, only slightly modifying the design variables.) This final mass compares well with the best-performing EVEJP trajectory from Yam et al. [85], which possesses a final mass of 10,449 kg and a TOF of 11.4 years. However, in Ref. 85, trajectories with flight times longer than 11.4 years were not examined. The characteristics of the best trajectory from the GA-GALLOP hybrid algorithm and the encounter dates corresponding to the 11.4 year trajectory from Ref. 85 are compared in Table 4.9. Beyond the dissimilar flight times, the solution from the GA-GALLOP hybrid is a different type of trajectory from the trajectory solution in Ref. 85. The trajectory from the GA-GALLOP hybrid originates from a launch date that is three months earlier than the trajectory in Ref. 85 and possesses a Venus flyby date 14 months later than the

case from Ref. 85. However, the phasing of the Earth and Jupiter flybys is similar. The best trajectory (in terms of final mass) from the hybrid algorithm is plotted in Fig. 4.12. The trajectory completes 1.5 solar revolutions before encountering Venus. The gravity assist from Venus then accelerates the spacecraft to a flyby with Earth only two months later. This Venus flyby places the spacecraft into a 2:2 transfer resonance[1] with Earth, which maximizes efficiency by avoiding costly periapsis rotation. Note that the spacecraft is thrusting along most of the E-V and V-E legs of the trajectory. Additionally, the spacecraft passes inside the orbit of Venus on the V-E leg, which may expose the spacecraft to high levels of radiation (the solar distance of the spacecraft was not constrained in the hybrid formulation). The GA-GALLOP hybrid algorithm also generates other high-performing options for the EVEJP example; the characteristics of two of these alternatives are described in Table 4.10. The second alternative possesses a TOF of 11.3 years, but with final mass of only 9411 kg, 1221 kg less than the apparent global optimum. Recall, however, that TOF is not an optimization objective.

[1]Resonance here means, *number of planet revolutions* : *number of spacecraft revolutions*.

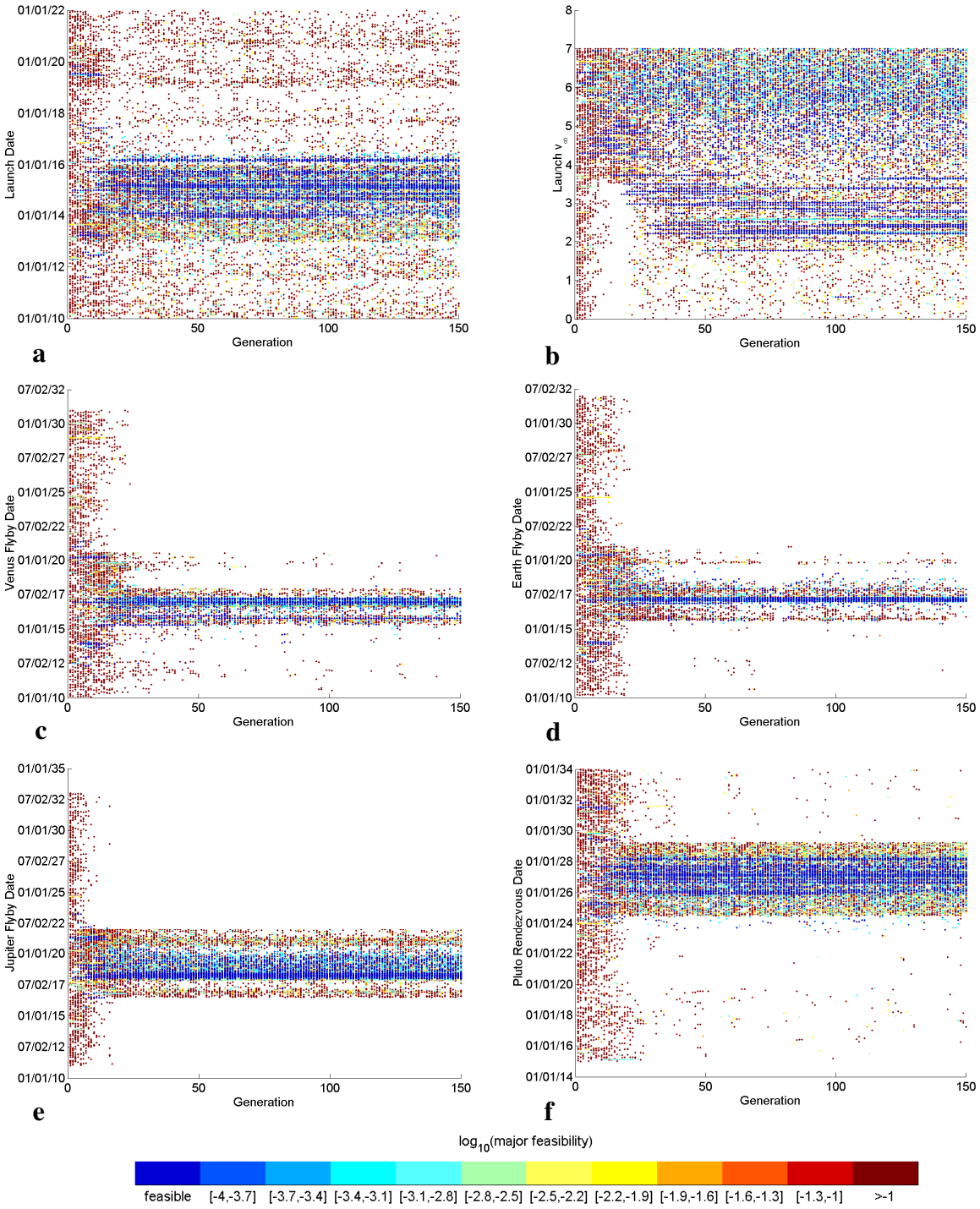

Figure 4.8. Evolution of key design variables for EVEJP rendezvous case (TOF ≤ 12 yrs). Dark blue indicates a feasible design. a) Earth launch date vs. generation number. b) Launch v_∞ vs. generation number. c) Venus flyby date vs. generation number. d) Earth flyby date vs. generation number. e) Jupiter flyby date vs. generation number. f) Pluto rendezvous date vs. generation number.

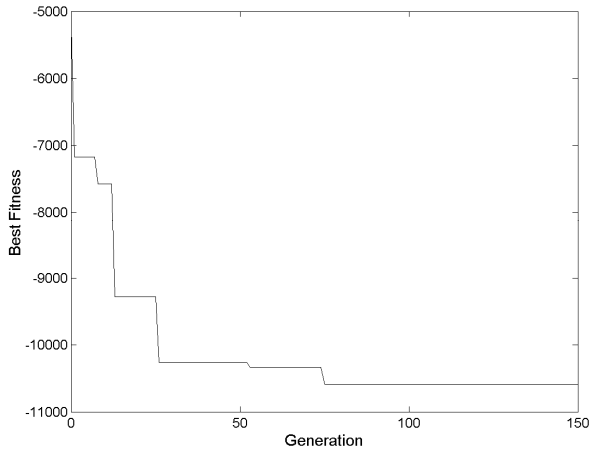

Figure 4.9 Fitness of best design through 150 generations for EVEJP example

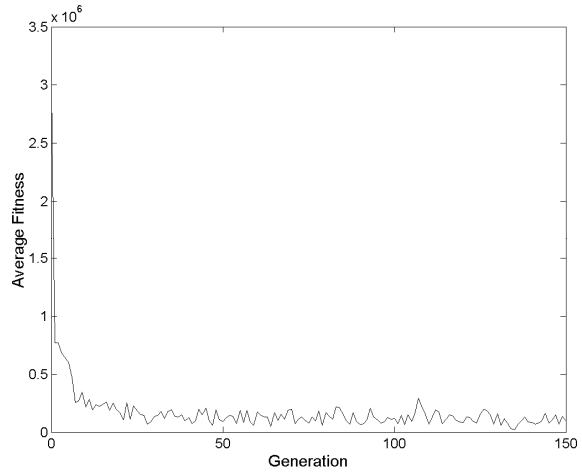

Figure 4.10 Average fitness of population through the evolution of the EVEJP example

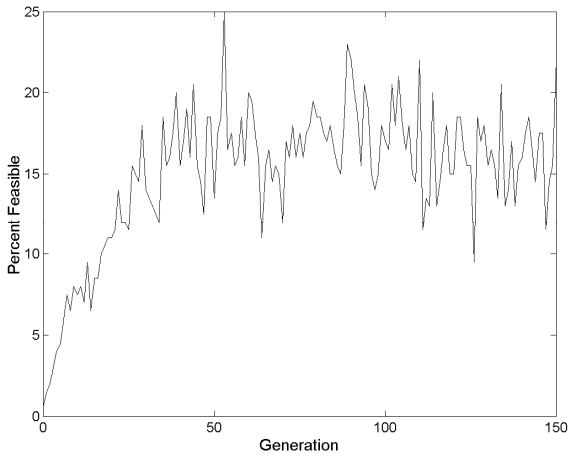

Figure 4.11 Percentage of feasible designs through the evolution of the EVEJP example

Table 4.9 Comparison of EVEJP rendezvous trajectory characteristics

Characteristics	GA-GALLOP Hybrid Value	Ref. 85 Value
Launch date	31 Jan. 2015	25 Apr. 2015
Launch v_∞	2.22 km/s	2.01 km/s
Injected mass	18,295 kg	18,591 kg
Venus flyby date	12 Dec. 2016	11 Oct. 2015
Venus flyby v_∞	15.51 km/s	N/A
Venus flyby altitude	500 km[a]	N/A
Venus flyby mass	16,162 kg	N/A
Venus flyby B-plane angle	-170[o]	N/A
Earth flyby date	11 Feb. 2017	24 Jan. 2017
Earth flyby v_∞	18.78 km/s	N/A
Earth flyby altitude	500 km[a]	N/A
Earth flyby mass	15,946 kg	N/A
Earth flyby B-plane angle	-12 [o]	N/A
Jupiter flyby date	21 Apr. 2018	6 May 2016
Jupiter flyby v_∞	15.96 km/s	N/A
Jupiter flyby altitude	164,739 km	N/A
Jupiter flyby mass	15,964 kg	N/A
Jupiter flyby B-plane angle	-5.19 [o]	N/A
Pluto arrival date	28 Jan. 2027	4 Sept. 2026
TOF	11.99 years[b]	11.4 years
Final mass	10,632 kg	10,449 kg

[a] on lower bound; [b] on upper bound

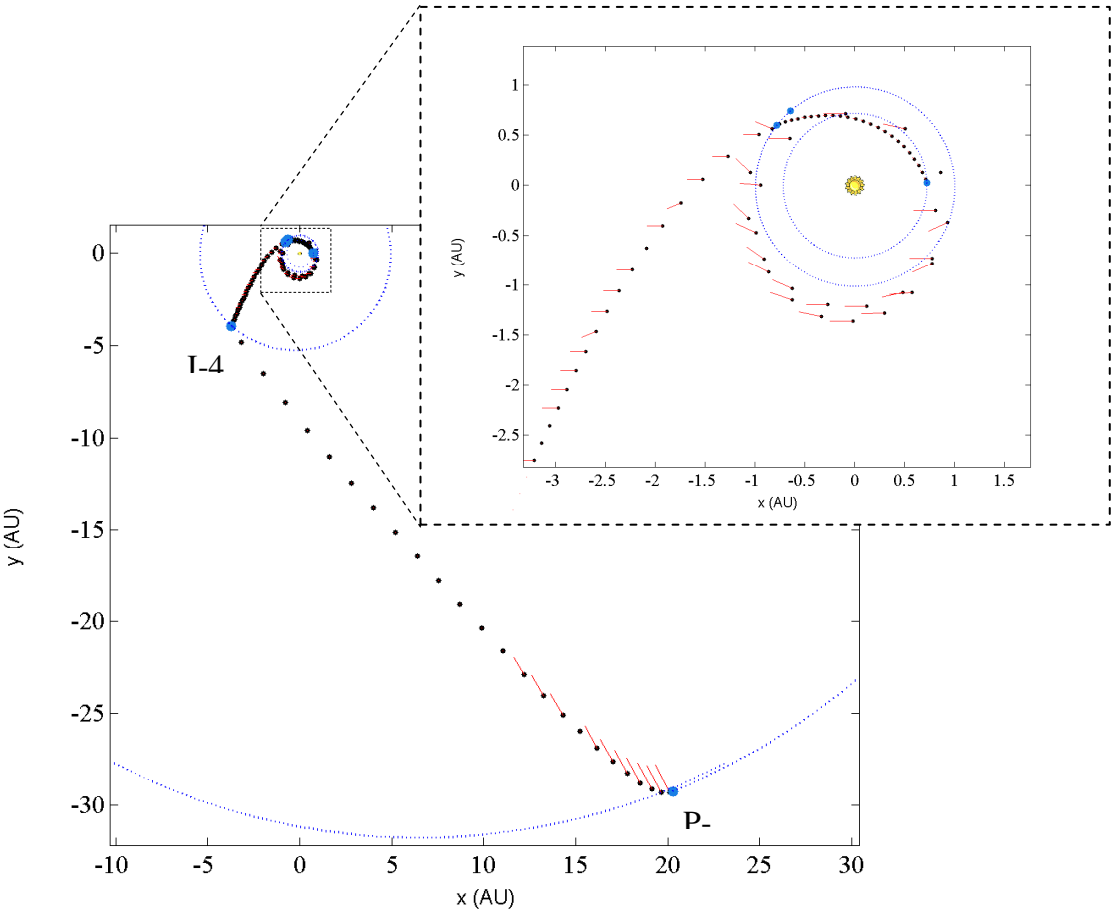

Figure 4.12. EVEJP rendezvous trajectory from GA-GALLOP hybrid (m_f=10,632 kg).

Table 4.10 High-performing, locally-optimal, EVEJP trajectories from GA-GALLOP hybrid algorithm

Launch Date	Launch v_∞ (km/s)	Flight times for each leg (days)	TOF (years)	Final mass (kg)
6 Sept. 2014	2.21	818, 69, 410, 3082	11.99	10,263
6 Feb. 2015	2.86	664, 71, 429. 2956	11.28	9,411

4.1.3 NEP Earth-Mars-Earth-Jupiter-Pluto Rendezvous

To further examine the capabilities of the GA-GALLOP hybrid, another rendezvous trajectory to Pluto is selected for comparison with Yam et al. [85]. Instead of employing Venus for an initial gravity assist, Mars is exploited, so that the trajectory sequence is EMEJP. The same spacecraft parameters (Table 4.6) are defined and many of the same GA-GALLOP hybrid parameters (Table 4.11), as in the EVEJP example, are used. The population size remains at 200 members and 7% elitism is again applied. However, in this new example, the major iterations limit is decreased to 600 and the Lamarckian buffer is reduced to 1E-4. Additionally, the lower and upper bounds on the launch date are modified to 2014 and 2025, respectively, to match the range in Ref. 85. The bounds on the other key design variables for this case are listed in Table 4.12. As in the EVEJP example, there are 171 design variables; however, the chromosome length is slightly longer at 988 bits.

Table 4.11 GA-GALLOP hybrid parameters for EMEJP rendezvous case

Parameter	Value
Population size	200
Elitism percentage	7%
Mutation probability	3%
Lamarckian probability	100%
Lamarckian buffer	1E-4
Maximum generation	150
Penalty parameter	1E8
Major iterations limit	600
Major feasibility tolerance	1E-4
Major optimality tolerance	1E-4
Degree of Chebyshev series	5

Table 4.12 GA-GALLOP hybrid key design variable bounds for EMEJP rendezvous case

Design Variable	Lower Bound	Upper Bound
Launch date	1 Jan. 2014	1 Jan. 2025
Launch v_∞ [km/s]	0	7
Mars arrival date	1 Feb. 2014	1 Jan. 2034
Mars flyby altitude [km]	500	10,000
Earth arrival date	1 Mar. 2014	1 Jan. 2035
Earth flyby altitude [km]	500	10,000
Jupiter arrival date	1 Jan. 2015	1 Jan. 2036
Jupiter flyby altitude [R_J][a]	2	40
Pluto arrival date	1 Jan. 2019	1 Jan. 2037
Final mass [kg]	2000	20,000

As in the previous two examples, to demonstrate the evolution of key design variables for the EMEJP case, the launch and encounter dates for the entire population are plotted versus the generation number in Fig. 4.13. Through 150 generations, the GA-GALLOP hybrid does not converge on a single design; however, bands of feasibility again form in each of the plots. Except for the evolution of the Earth flyby date, narrow regions of encounter dates do not emerge as in the EVEJP example. This difference may be the result of the more selective application of Lamarckian inheritance to only feasible solutions from GALLOP in combination with the lower maximum iteration limit. Like the previous EVEJP example, the majority of the designs throughout the evolution possess a high launch v_∞, especially in the early generations. The best performing trajectories, however, eventually reach a relatively low launch v_∞. Thus, encouraging low launch v_∞ in future implementations may improve efficiency. The evolution of the best fitness, average fitness value, and percent feasible are plotted in Figs. 4.14 through 4.16. As in the EVEJP example, the best fitness decreases quickly in the initial generations and then plateaus. Contrastingly, however, significant improvements are realized in the later generations, after ~80 generations of consistent values of best fitness. The evolution of the average fitness value and the percent of the solutions that are feasible follow patterns similar to the corresponding plots in the EVEJP example, undergoing dramatic changes

in the first few evolutions, then fluctuating within a narrower range. The overall trend in terms of the percentage of trajectories that are feasible is increasing, a desirable outcome from an efficiency standpoint.

The best solution generated by the GA-GALLOP hybrid algorithm (generation 138) is highlighted by a final mass of 10,339 kg and a time-of-flight of 11.99 years (after re-optimization with "major feasibility" and "major optimality" reduced to 1E-6). This is a ~13% increase in final mass compared to the best EMEJP solution in Yam et al. [85], which is defined by a final mass of 9,162 kg and a TOF of 11.7 years. The trajectory from the GA-GALLOP hybrid algorithm possesses a higher launch v_∞ and a slightly longer TOF compared to the trajectory from Ref. 85. The characteristics of the two trajectories are displayed in Table 4.13. Similar to the previous EVEJP best trajectory, a resonance for the Earth return is again apparent. For this example, a trajectory with a 2:1 resonance is identified as illustrated in Fig. 4.17. The Mars flyby efficiently raises apoapsis, compensating for the relatively high launch v_∞ required for the inner planet phasing. Note that the trajectory does pass inside Earth's orbit, which may not be desirable, but was not a constraint. Other high-performing trajectories are also identified in this hybrid run, including a trajectory with a final mass of 9,648 kg and TOF of 11.7 years as well as a trajectory with a final mass of 9,167 kg and a TOF of 10.8 years (Table 4.14).

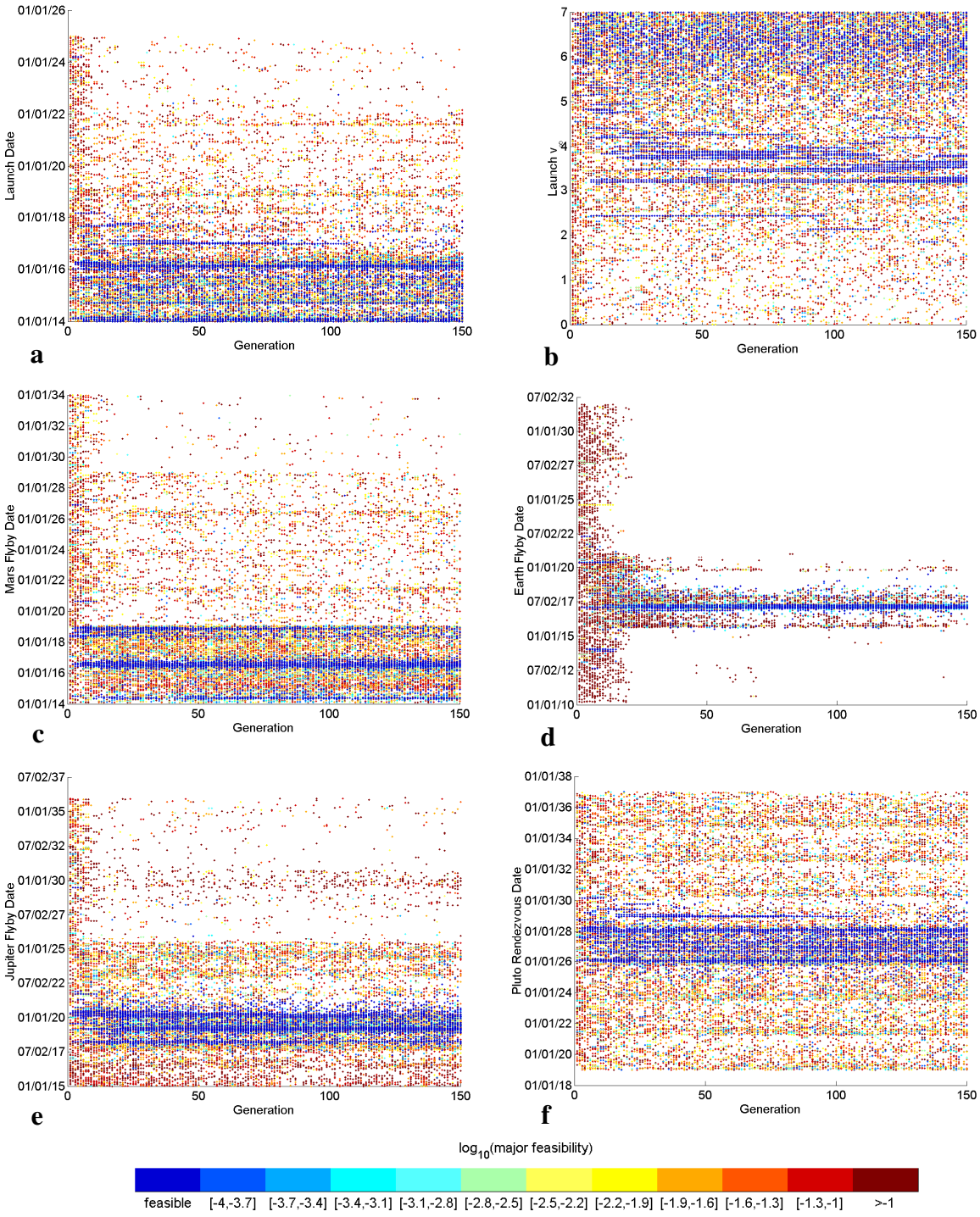

Figure 4.13. Evolution of key design variables for EMEJP rendezvous case. Dark blue indicates a feasible design. a) Earth launch date vs. generation number. b) Launch v_∞ vs. generation number. c) Mars flyby date vs. generation number. d) Earth flyby date vs. generation number. e) Jupiter flyby date vs. generation number. f) Pluto rendezvous date vs. generation number.

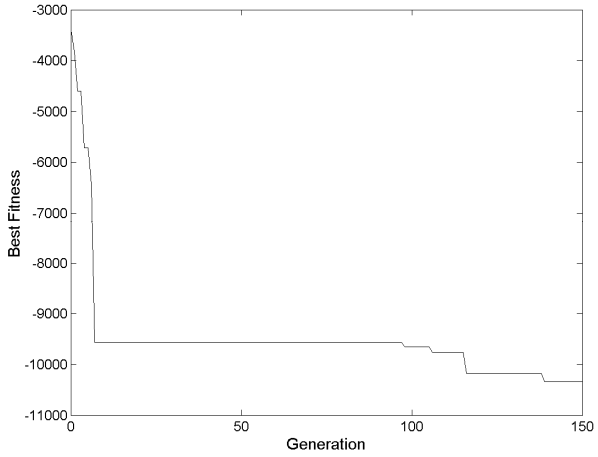

Figure 4.14 Fitness of best design through 150 generations for EMEJP example

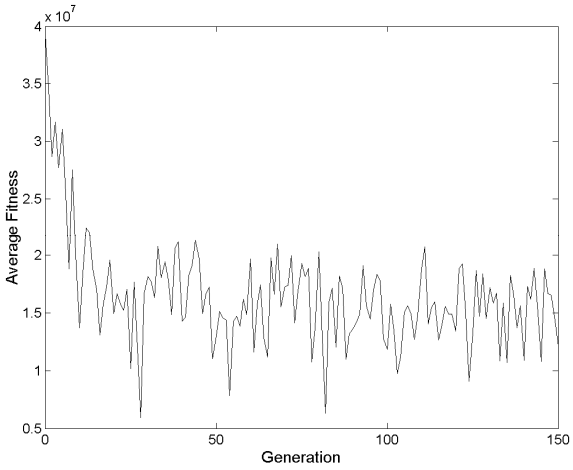

Figure 4.15 Average fitness of population through the evolution of the EMEJP example

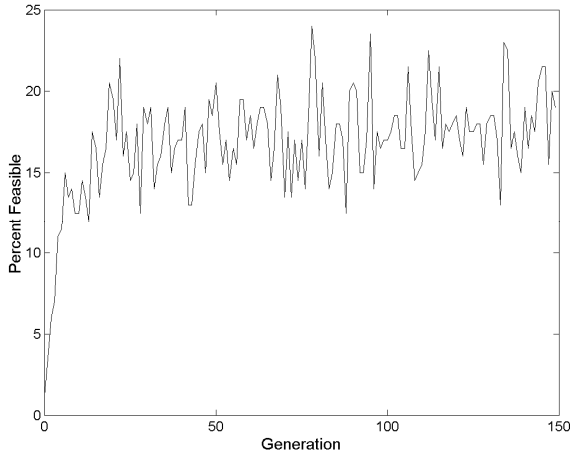

Figure 4.16 Percentage of feasible designs through the evolution of the EMEJP example

Table 4.13 EMEJP rendezvous trajectory characteristics

Characteristics	GA-GALLOP Hybrid Value	Ref. 85 Value
Launch date	14 Feb. 2016	19 Mar. 2014
Launch v_∞	3.26 km/s	2.06 km/s
Injected mass	16,464 kg	N/A
Mars flyby date	4 July 2016	20 Mar. 2016
Mars flyby v_∞	10.38 km/s	N/A
Mars flyby altitude	712 km	N/A
Mars flyby mass	15,993 kg	N/A
Mars flyby B-plane angle	-21^o	N/A
Earth flyby date	30 Dec. 2017	30 Dec. 2016
Earth flyby v_∞	11.75 km/s	N/A
Earth flyby altitude	500 km[a]	N/A
Earth flyby mass	14,640 kg	N/A
Earth flyby B-plane angle	180^o	N/A
Jupiter flyby date	10 Mar. 2019	14 Apr. 2018
Jupiter flyby v_∞	17.89 km/s	N/A
Jupiter flyby altitude	670,352 km	N/A
Jupiter flyby mass	13,999 kg	N/A
Jupiter flyby B-plane angle	-9^o	N/A
Pluto arrival date	10 Feb. 2028	12 Dec. 2025
TOF	11.99 years[b]	11.7
Final mass	10,339 kg	9,162 kg

[a] on lower bound; [b] on upper bound

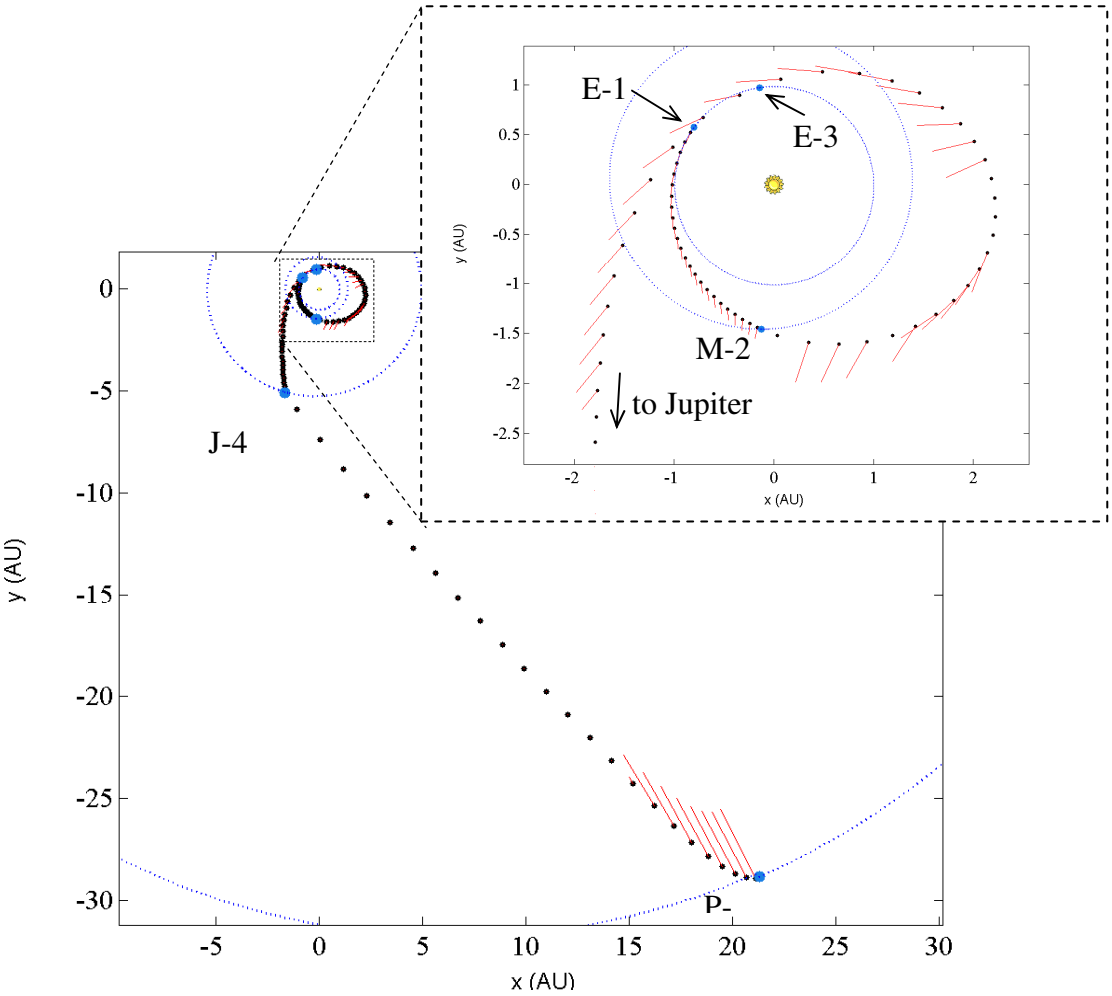

Figure 4.17. EMEJP rendezvous trajectory from GA-GALLOP hybrid (m_f=10,339 kg)

Table 4.14 High-performing, locally-optimal, EMEJP trajectories from GA-GALLOP hybrid algorithm

Launch Date	Launch v_∞ (km/s)	Flight times for each leg (days)	TOF (years)	Final mass (kg)
15 Feb. 2016	3.53	137, 541, 438, 3154	11.69	9,648
14 Feb. 2016	3.40	139, 541, 422, 3029	11.31	9,305
13 Feb. 2016	3.21	142, 539, 411, 2846	10.79	9,166

4.1.4 NEP Earth-Venus-Earth-Jupiter-Neptune Rendezvous

The GA-GALLOP hybrid algorithm is next applied to an EVEJN rendezvous example. Again, the example is compared to Yam et al. [85] to ensure that the known optimum is met or exceeded. The spacecraft parameters are consistent with the previous EVEJP and EMEJP examples; the same GA-GALLOP hybrid parameters as those in the EMEJP case are also used, except the major iterations limit, which is increased to 700 (Table 4.15). The bounds on the primary design variables are listed in Table 4.16 where the launch date range is the same as in Ref. 85. The number of design variables and chromosome length are the equal to those in the EMEJP example, at 171 and 988 respectively.

Table 4.15 GA-GALLOP hybrid key design variable bounds for EVEJN rendezvous case

Design Variable	Lower Bound	Upper Bound
Launch date	1 Jan. 2014	1 Jan. 2025
Launch v_∞ [km/s]	0	7
Mars arrival date	1 Feb. 2014	1 Jan. 2034
Mars flyby altitude [km]	500	10,000
Earth arrival date	1 Mar. 2014	1 Jan. 2035
Earth flyby altitude [km]	500	10,000
Jupiter arrival date	1 Jan. 2015	1 Jan. 2036
Jupiter flyby altitude [R_J][a]	2	40
Pluto arrival date	1 Jan. 2019	1 Jan. 2037
Final mass [kg]	2000	20,000

Table 4.16 GA-GALLOP hybrid parameters for EVEJN rendezvous case

Parameter	Value
Population size	200
Elitism percentage	7%
Mutation probability	3%
Lamarckian probability	100%
Lamarckian buffer	1E-4
Maximum generation	150
Penalty parameter	1E8
Major iterations limit	700
Major feasibility tolerance	1E-4
Major optimality tolerance	1E-4
Degree of Chebyshev series	5

The evolution of the key design variables is illustrated in Fig. 4.18. By the 150th generation, a wide band of feasible launch dates is still apparent. However, between generation 40 and generation 60, the focus of the search shifts from an earlier band of feasible launch dates (2015 to 2017) to a later band (2017 to 2020). Relatively narrow bands of feasibility develop for the Venus and Earth flyby dates, whereas the feasible ranges for the optimal Jupiter flyby date and Neptune arrival date are wide. Again, trajectories with a low launch v_∞ are not feasible in the early generations, but incrementally emerge through the evolution as feasible trajectories with smaller launch v_∞ are identified. The evolution of the best fitness, average fitness, and percentage of the population that is feasible is plotted in Figs. 4.19 to 4.21. Each plot follows similar trends to the corresponding plots in the EVEJP and EMEJP examples.

The GA-GALLOP hybrid algorithm generates a trajectory with a final mass that is higher than the result reported in Yam et al. [85]. As with the other single objective examples, the solution is re-optimized in GALLOP with “major feasibility” and “major optimality” tolerances reduced to 1E-6. A trajectory with a final mass of 12,786 kg and a time-of-flight on the upper limit of 12 years is identified by the hybrid algorithm in the

85th generation. This is a ~9% increase over the optimal 11.8 year EVEJN trajectory in Ref. 85, which has a final mass of 11,783 kg. It is not clear if the TOF was constrained in the analysis in Ref. 85 or if certain trajectory characteristics were favored. A comparison of the trajectory characteristics of the two solutions appears in Table 4.17. The best trajectory from the hybrid launches a few months earlier, and with a lower launch v_∞, than the corresponding solution from Ref. 85. The main difference between the two trajectories occurs in the inner planet phasing, whereas the Jupiter and Neptune arrival dates of the two solutions are similar. The best performing trajectory from the hybrid algorithm is plotted in Fig. 4.22. Note the similarities between the geometry of the EVE sequence in this trajectory and that from the EVEJP case. Again, a 2:2 transfer resonance with Earth is generated, possible with the short V-E leg. As with the EVEJP example, the trajectory again passes inside the orbit of Venus. Alternative trajectories, which possess lower final mass values than the apparent global optimum, are detailed in Table 4.18.

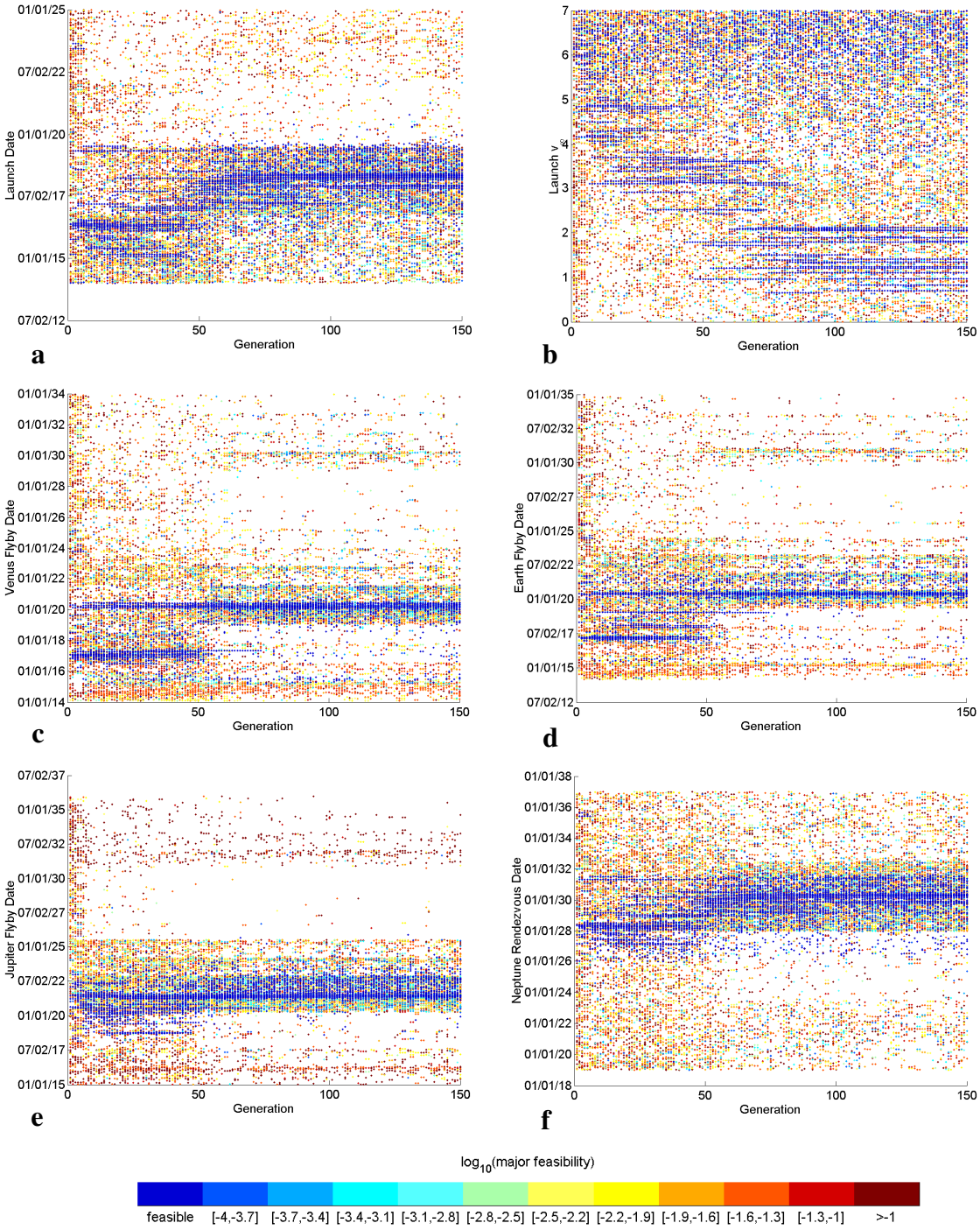

Figure 4.18. Evolution of key design variables for EVEJN rendezvous case. Dark blue indicates a feasible design. a) Earth launch date vs. generation number. b) Launch v_∞ vs. generation number. c) Venus flyby date vs. generation number. d) Earth flyby date vs. generation number. e) Jupiter flyby date vs. generation number. f) Pluto rendezvous date vs. generation number.

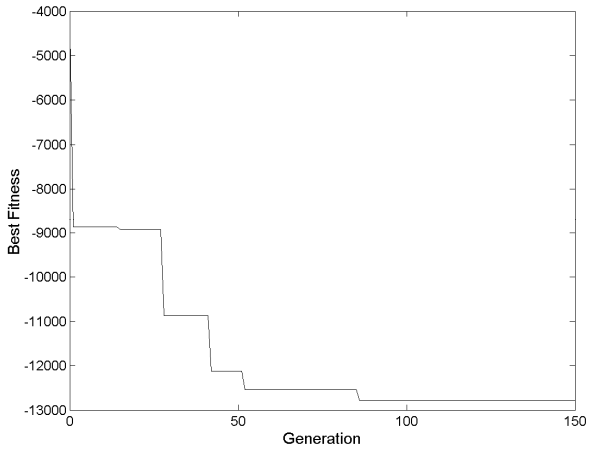

Figure 4.19 Fitness of best design through 150 generations for EVEJN example

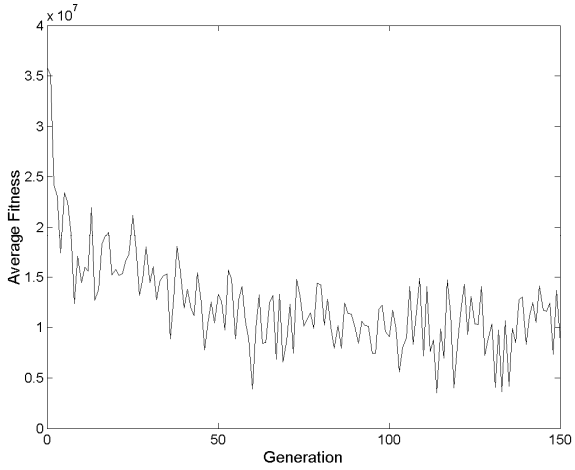

Figure 4.20 Average fitness of population through evolution of EVEJN example

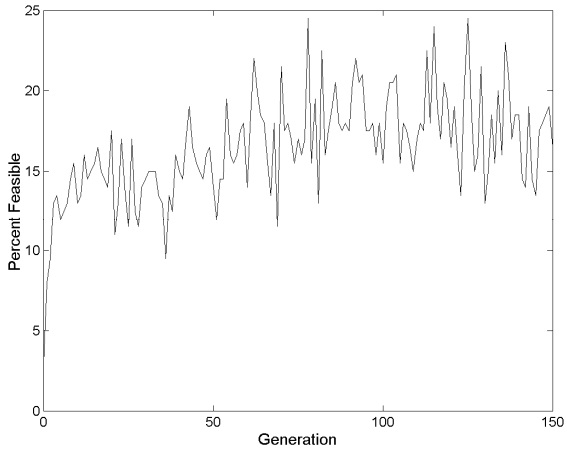

Figure 4.21 Percentage of feasible designs through evolution of EVEJN example

Table 4.17 EVEJN rendezvous trajectory characteristics

Characteristics	GA-GALLOP Hybrid Value	Ref. 85 Value
Launch date	10 Apr. 2018	26 Aug. 2018
Launch v_∞	1.23 km/s	2.22 km/s
Injected mass	19,458 kg	N/A
Venus flyby date	7 Mar. 2020	1 Mar. 2019
Venus flyby v_∞	12.64 km/s	N/A
Venus flyby altitude	500 km	N/A
Venus flyby mass	17,435 kg	N/A
Venus flyby B-plane angle	170^o	N/A
Earth flyby date	13 May 2020	12 Feb. 2020
Earth flyby v_∞	17.52 km/s	N/A
Earth flyby altitude	500 km	N/A
Earth flyby mass	17,231 kg	N/A
Earth flyby B-plane angle	5^o	N/A
Jupiter flyby date	29 May 2021	14 July 2021
Jupiter flyby v_∞	16.82 km/s	N/A
Jupiter flyby altitude	1,133,457 km	N/A
Jupiter flyby mass	16,611 kg	N/A
Jupiter flyby B-plane angle	-3^o	N/A
Neptune arrival date	7 Apr. 2030	4 June 2030
TOF	11.99 years	11.8
Final mass	12,786 kg	11,783 kg

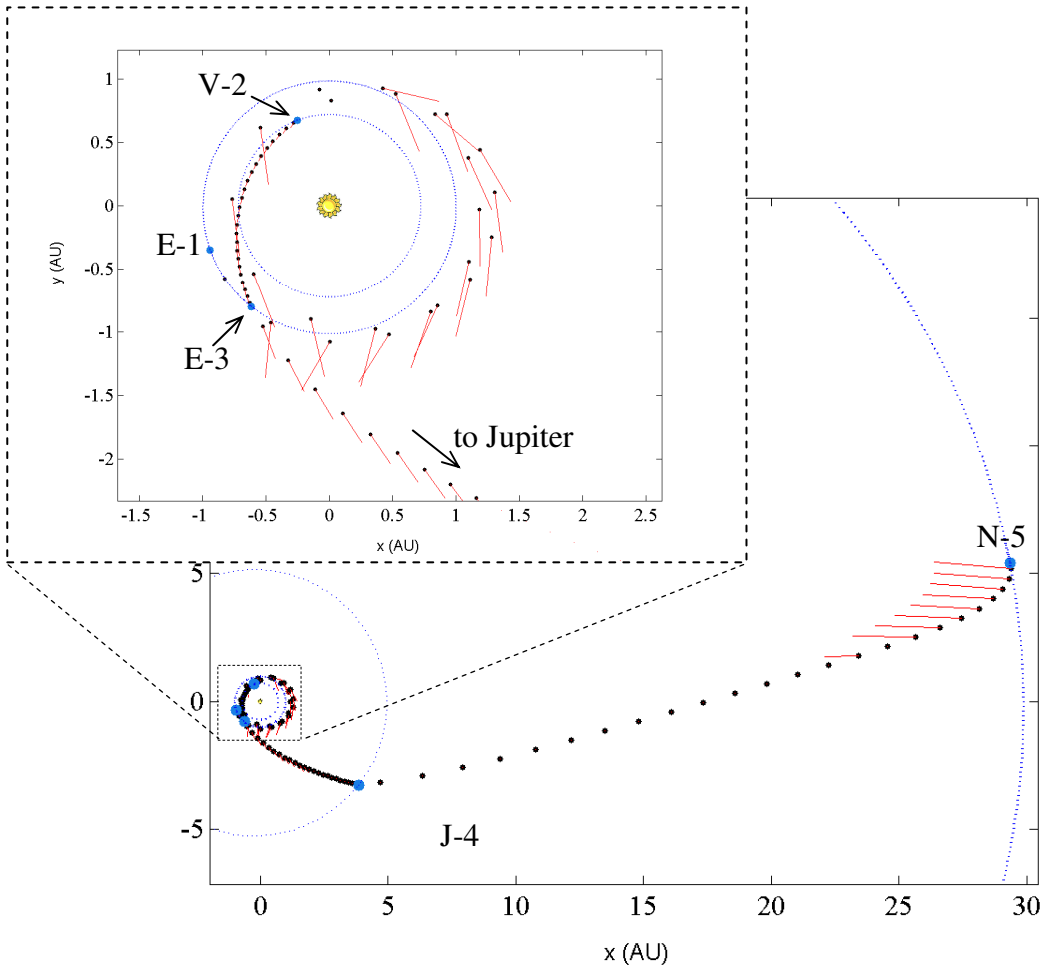

Figure 4.22. EVEJN rendezvous trajectory from GA-GALLOP hybrid (m_f=12,786 kg)

Table 4.18 High-performing, locally optimal, EVEJN trajectories from the GA-GALLOP hybrid

Launch Date	Launch v_∞ (km/s)	Flight times for each leg (days)	TOF (years)	Final mass (kg)
7 Dec. 2017	0.83	826, 63, 371, 3119	11.99	12,519
2 June 2018	2.05	645, 66, 383, 3286	11.99	12,315
11 May 2017	0.80	1032, 65, 362, 2921	11.99	12,086

4.2 Multiobjective Hybrid Results

The multiobjective implementation of the GA-GALLOP hybrid algorithm is applied to a direct Earth-to-Mars rendezvous mission scenario with the goal of generating a Pareto front comprising trajectories that are optimal in terms of final mass and time-of-flight. To verify the capability of the hybrid to develop a representative front of globally optimal solutions in terms of both objectives, the same solar electric propulsion powered spacecraft (Table 4.1), launch vehicle (Table 4.2), and date ranges (Table 4.4) as in the single objective Earth-Mars example of Section 4.1 are employed. Globally optimal solutions in terms of final mass are available for flight times of 1.5, 2.0, 2.5, 3.0, and 3.5 years; these trajectories are apparent from the peaks of the contours in Figure 4.6 (based on a plot from Williams and Coverstone-Carroll [84]) and were corroborated with the single objective implementation of the GA-GALLOP hybrid algorithm. Thus, these five points are located along the Pareto front and serve as test points to validate that the hybrid algorithm is able to represent the Pareto front adequately. The launch and arrival dates (as with the rest of the design variables) are not fixed, and are allowed to vary within the specified ranges. The time-of-flight is constrained to be less than 3.5 years, and the population size is 200 (an increase of 100 from the corresponding single objective example). The hybrid parameters are listed in Table 4.19. A comprehensive study was not conducted to develop optimal hybrid parameter values and, thus, a larger population size may be beneficial. Note that the specification of an elitism percentage is not required because the NSGA-II inherently incorporates elitism.

Table 4.19 Multiobjective GA-GALLOP hybrid parameters for EM rendezvous case

Parameter	Value
Population size	200
Mutation probability	3%
Lamarckian probability	100%
Lamarckian buffer	1E-4
Maximum generation	800
Penalty parameter	1E8
Major iterations limit	300
Major feasibility tolerance	1E-4
Major optimality tolerance	1E-4
Degree of Chebyshev series	4

To demonstrate the evolution of the population in the multiobjective GA-GALLOP hybrid algorithm, both objectives are plotted against each other for generations 1, 3, 10, 25, 50, 100, 200, and 800 in Fig. 4.23. In the first two generations, several individuals in the population are infeasible, and, therefore, both objectives are penalized, rendering the low fitness values in generation 1. However, by the third generation, all of the trajectories in the population are feasible, though many are clustered near the TOF upper limit (3.5 years). By the 10^{th} generation, the multiobjective GA-GALLOP hybrid algorithm has discovered several more trajectories with lower flight times that are non-dominated. This trend continues through the 25^{th} generation, where a more even spread of non-dominated trajectories has been generated. At generation 100, most of the population lies along a distinct non-dominated front. From generation 100 to 800, this front is progressively shifted up (higher final mass) and to the left (lower flight time), towards the Pareto front. By the 800^{th} generation, all members of the population are non-dominated and distributed uniformly along the front.

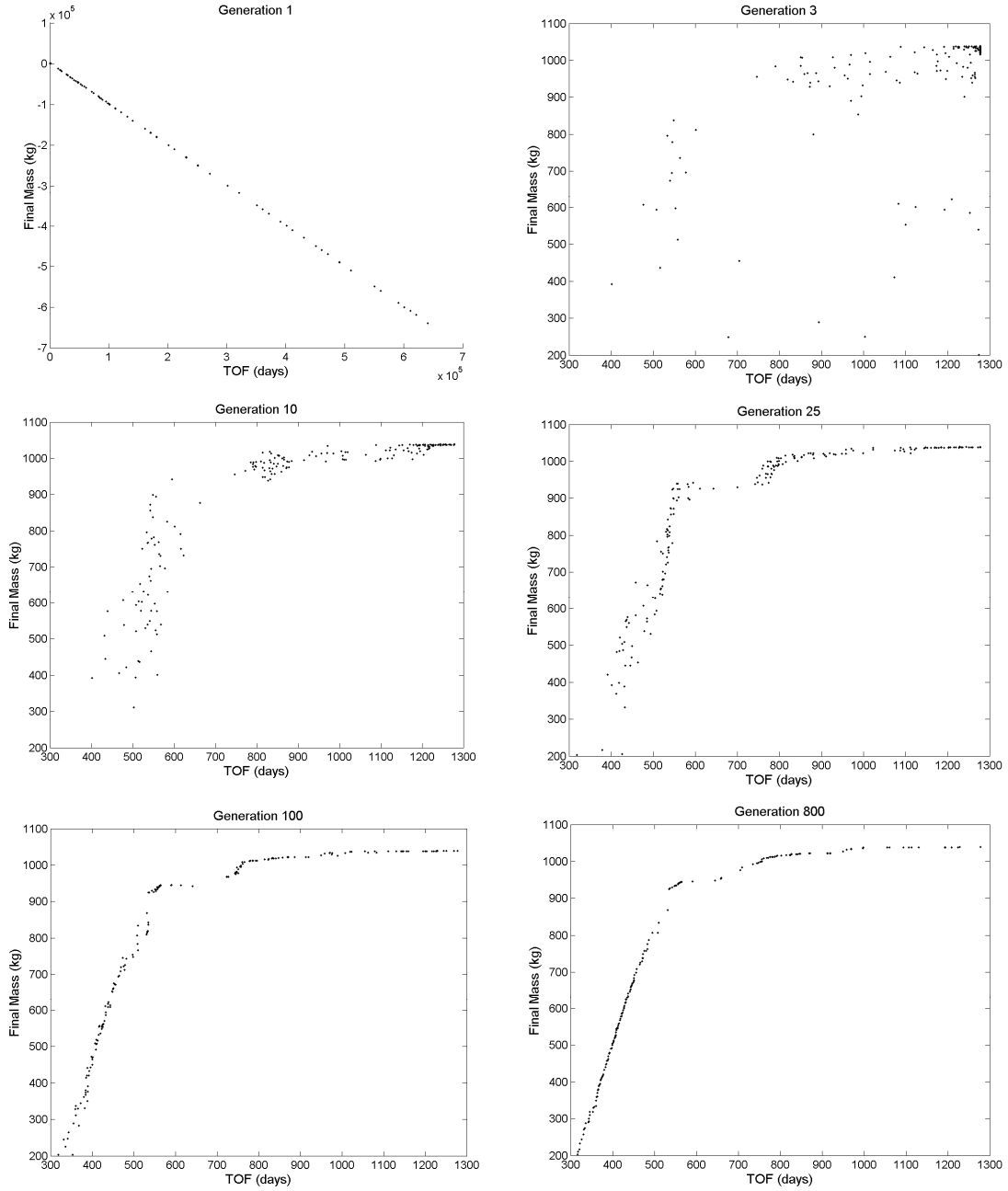

Figure 4.23 Evolution of the population for multiobjective EM rendezvous from GA-GALLOP hybrid (generations 1, 3, 10, 25, 100, and 800)

The final mass versus the time-of-flight for all of the feasible designs throughout the evolution (800 generations) are plotted in Fig. 4.24. The trajectories corresponding to the final generation are identified in red, whereas the rest of the designs are blue. Note that four clusters of trajectories have developed in different TOF regions. One family resides in the TOF region from 300 days to roughly 510 days, where relatively dramatic improvements in final mass can be realized with only small compromises in TOF. The next family stems from the disjointed shoulder of the non-dominated front from ~510 days to ~570 days. The third family is the grouping of trajectories with flight times from 700 days to about 950 days, where the front begins to level out, and only small improvements in final mass are attained with increasing TOF. The final family is located in the upper right hand corner, with TOFs ranging from ~1050 days to 1278 days. The trajectories in each family possess similar characteristics and thrusting structures (e.g., coast-thrust or thrust-coast-thrust), where the trajectories with longer flight times and higher mass within each family, generally, launch earlier, with a lower v_∞, and arrive later.

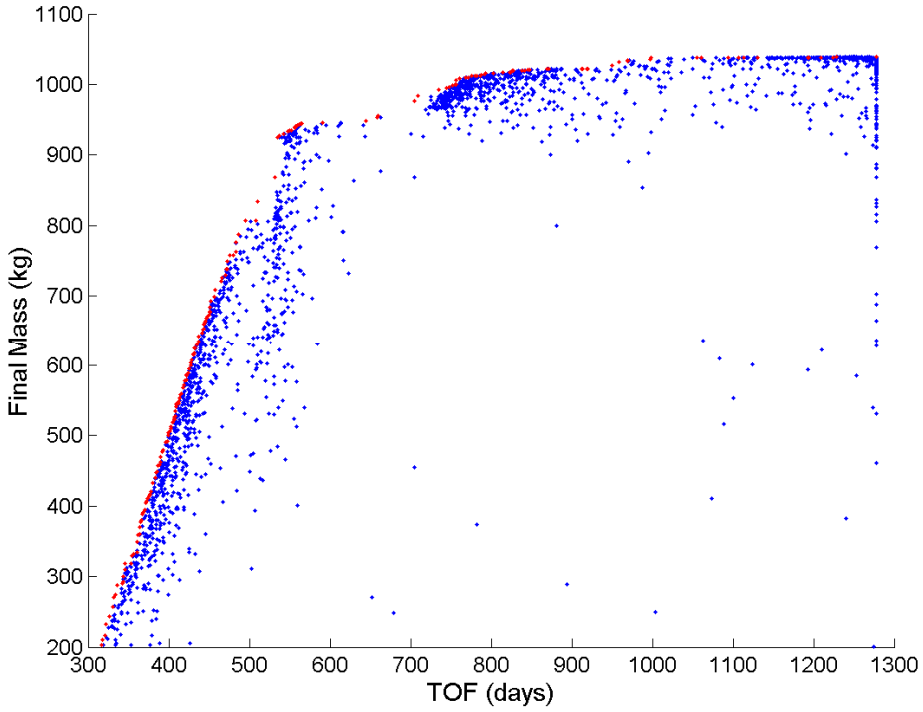

Figure 4.24 All feasible designs through 800 generations in the multiobjective EM rendezvous example (designs from generation 800 in red)

After 800 generations, each member of the final population is sent to GALLOP to be re-optimized with "major feasibility" and "major optimality" tolerances both decreased from 1E-4 to 1E-6, and the flight time constrained to its current value. Recall that GALLOP optimizes for final mass only, and in the nominal execution of the multiobjective GA-GALLOP hybrid, the TOF is not constrained to a particular length; however, an upper limit of 3.5 years is in effect. The refined population is plotted in Fig. 4.25. Several of the individuals in the family, those with lower flight times, shift to solutions with a higher final mass. Correspondingly, the non-dominated front is also shifted, and as a result, the population no longer comprises a single set of non-dominated individuals. However, the non-dominated front associated with the re-optimized population should be more representative of the true Pareto front in the shorter TOF region. To verify that the non-dominated front is near the Pareto front, the point solutions from the peaks of the contours of Fig. 4.6 are plotted as blue circles in Fig. 4.25. The five test points, validated using the single objective GA-GALLOP hybrid, are: (1.5 years, 943 kg), (2.0 years, 1008 kg), (2.5 years, 1024 kg), (3.0 years, 1038 kg), (3.5 years, 1039 kg). The test points successfully lie along the non-dominated front generated by the multiobjective GA-GALLOP hybrid. However, no test points exist to confirm that the individuals in the early TOF family, those that shifted after re-optimization, are representative of the Pareto front.

To illustrate the characteristics of the four families, several representative trajectories from the non-dominated front are selected for plotting. The location of these trajectories along the front is indicated by the green dots and a corresponding identification letter (*a* – *l*) in Fig. 4.25. The 12 selected trajectories are plotted in Figs. 4.26 and 4.27, and the characteristics of each trajectory are listed in Table 4.20. Trajectories in the short TOF family (trajectories *a* – *d*) possess similar launch and arrival dates, as well as a similar thrusting structure, that is, thrust-coast-thrust. However, as the TOF increases, the launch v_∞ decreases, from 317 days and 8.69 km/s in trajectory *a* to 427 days and 4.6 km/s in trajectory *d*. The trajectories with a higher TOF are able to thrust longer, allowing the

propellant-efficient low-thrust engine to accumulate enough ΔV to compensate for the reduced launch v_∞, which, otherwise, must be generated by the less efficient launch vehicle. In trajectory *a*, the spacecraft is thrusting along approximately 50% of the trajectory, whereas in trajectory *d*, the s/c is thrusting for all but one segment. In the 'shoulder' family, which includes trajectories *e* and *f*, the launch date is in late-August 2007, and the arrival date is in early-2009. In both of these trajectories, and the family in general, the spacecrafts is thrusting for nearly the entire trajectory. However, the longer flight time of trajectory *f* allows for a reduced launch v_∞ and, thus, a higher injection mass and final mass. Trajectories *g* and *h* are members of the third family, in which the spacecraft completes at least one solar revolution, launching in mid-2009 and then spiraling out to Mars by thrusting in a direction that is nearly tangential to the velocity. While trajectories *g* and *h* possess a similar launch v_∞, the longer TOF associated with trajectory *h* allows for the elimination of thrusting in inefficient locations along the trajectory (i.e., when the spacecraft is further from the Sun). The fourth, and final family, includes trajectories *i* through *l*. These solutions are also spiral trajectories; however, they launch in 2008, and execute more solar revolutions than the trajectories that are members of the third family.

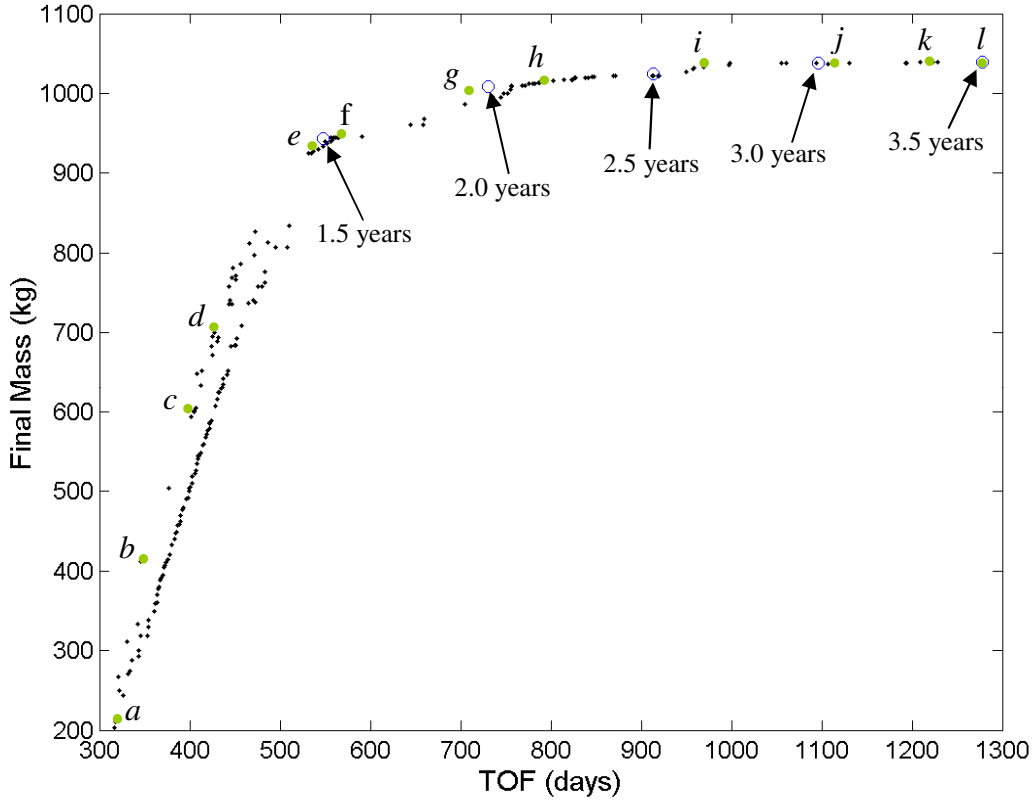

Figure 4.25 Re-optimized designs from generation 800 for multiobjective EM rendezvous example; blue circles indicate known optimums at fixed TOFs from Williams and Coverstone-Carroll [84]; trajectories selected for plotting are in green (a – l)

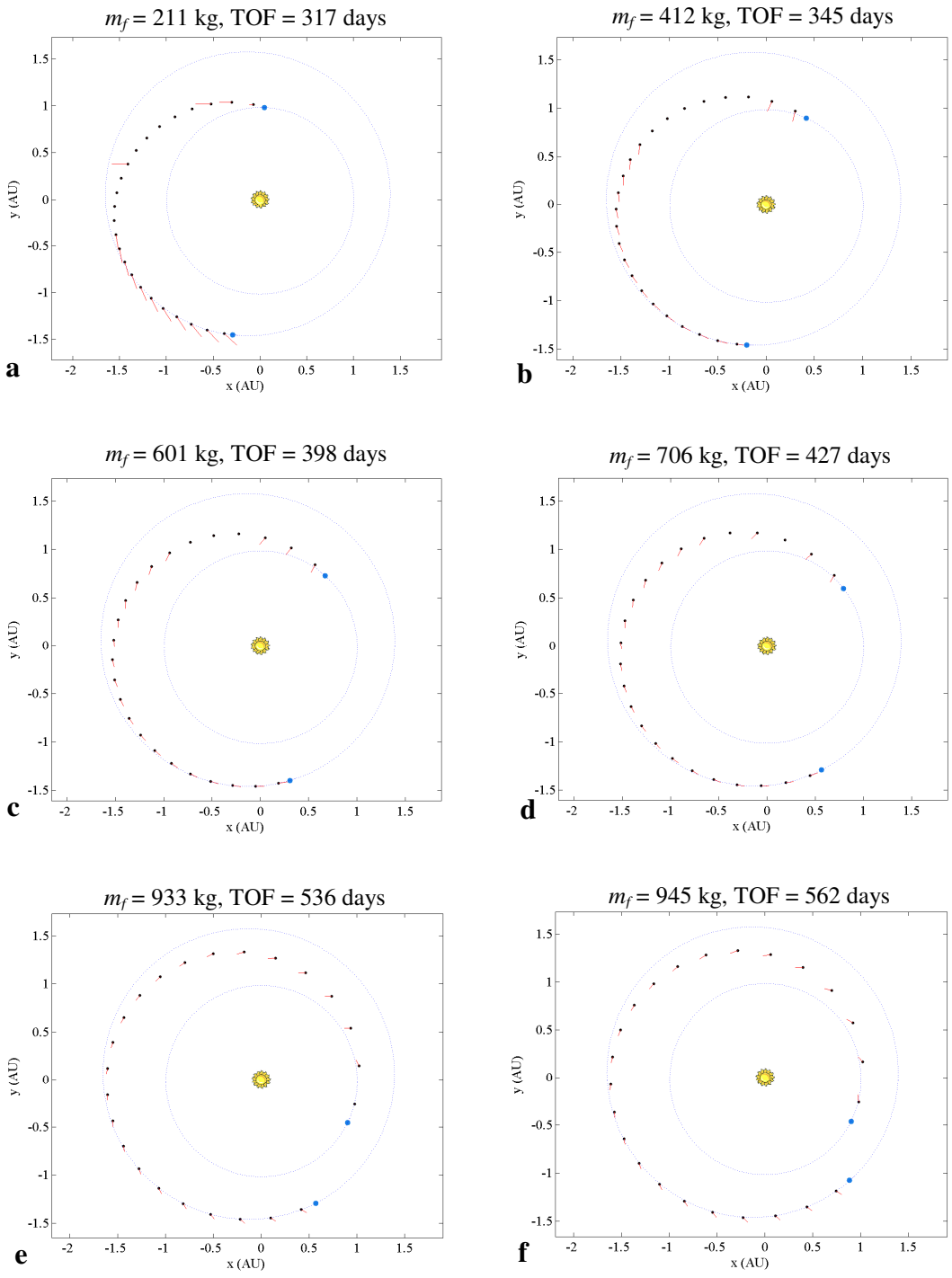

Figure 4.26 Ecliptic projections of selected trajectories from the re-optimized, generation 800 (trajectories *a* – *f*)

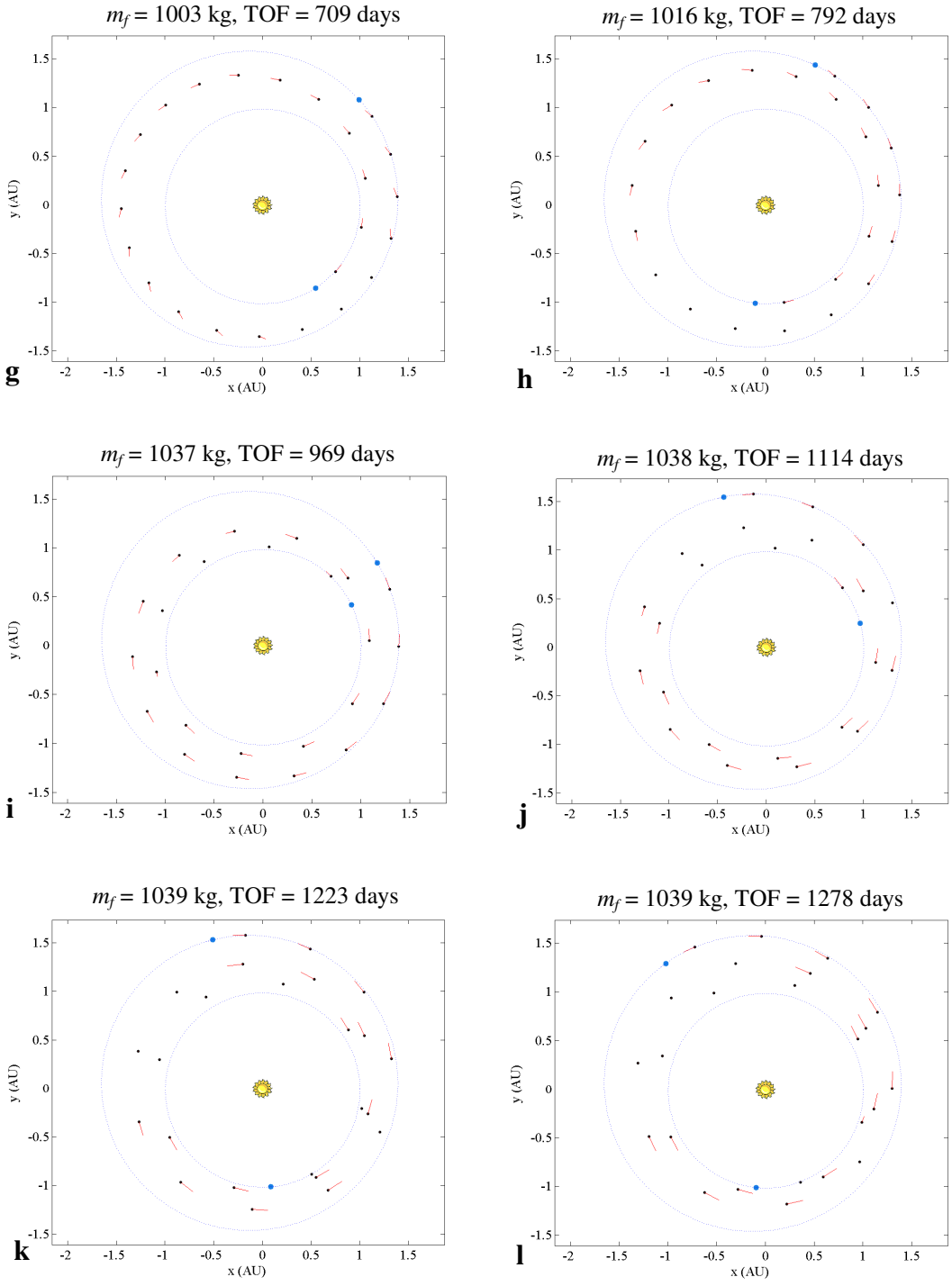

Figure 4.27 Ecliptic projections of selected trajectories from the re-optimized, generation 800 (trajectories *g* – *l*)

Table 4.20 Characteristics of selected trajectories (from Fig. 4.25) of multiobjective EM rendezvous example

Trajectory	Launch Date	Launch v_∞ (km/s)	Arrival Date	TOF (days)	Final Mass (kg)
a	19 Dec. 2009	8.69	1 Nov. 2010	317	211
b	27 Nov. 2009	6.86	7 Nov. 2010	345	412
c	9 Nov. 2009	5.41	12 Dec. 2010	398	601
d	30 Oct. 2009	4.66	31 Dec. 2010	427	706
e	27 Aug. 2007	2.90	12 Feb. 2009	536	933
f	26 Aug. 2007	2.71	10 Mar. 2009	562	945
g	25 July 2009	1.92	4 July 2011	709	1003
h	15 June 2009	1.73	15 Aug. 2011	792	1016
i	18 Oct. 2008	0.75	14 June 2011	969	1037
j	7 Oct. 2008	0.76	26 Oct. 2011	1114	1038
k	26 June 2008	0.75	1 Nov. 2011	1223	1039
l	15 June 2008	0.75	14 Dec. 2011	1278	1039

5. CONCLUSIONS

5.1 Summary

A hybridized technique for global low-thrust, gravity-assist trajectory optimization is developed. A genetic algorithm and the software GALLOP, which incorporates a calculus-based direct method, are combined to overcome many of the difficulties inherent to traditional LTGA optimization schemes. The structure of the hybrid approach allows an automated and efficient global exploration of the entire design space without the necessity of any predetermined initial guess. Biases associated with techniques that generate initial guesses based on simplifying assumptions are avoided, and nonintuitive optimums can be discovered. Both a single objective and multiobjective implementation of the hybrid algorithm are developed. In the single objective implementation, a simple genetic algorithm is incorporated as the global component of the hybrid, with the sole objective to maximize the spacecraft's final mass. Instead of a simple genetic algorithm, the multiobjective implementation incorporates the Non-dominated Sorting Genetic Algorithm II, so that a representation of the globally optimal Pareto front is generated in a single execution of the hybrid algorithm. Two objectives are considered: maximization of final mass and minimization of time-of-flight. However, both implementations can easily be modified for different objectives. Similarly, the hybrid algorithm is amenable to additional constraints, which can be readily incorporated into the genetic algorithm of the hybrid, without modifications to GALLOP.

The single objective implementation of the GA-GALLOP hybrid algorithm is applied to several LTGA trajectory examples with a relatively wide range of potential launch and encounter dates. A direct trajectory for a Mars rendezvous using solar electric propulsion is studied, along with nuclear electric propulsion examples that incorporate

multiple gravity assists to Pluto and Neptune. The final mass associated with the trajectories generated by the hybrid algorithm match or exceed those previously published for the examples examined. The technique is particularly effective for the development of high-performing solutions in complex, multiple gravity-assist scenarios. The execution of the algorithm does require several thousand individual local optimizations with GALLOP; however, the number of evaluations is significantly less than would be necessary with a grid search at a resolution that is sufficiently fine to capture all trajectory possibilities.

The multiobjective implementation is applied to the same Earth-Mars SEP design scenario as in the single objective implementation. A representative Pareto front of globally optimal trajectories in terms of final mass and time-of-flight is generated in one execution of the hybrid algorithm. This entire set of non-dominated solutions can be utilized to perform a trade study on the two objectives, significantly aiding the mission designer in the selection of a final trajectory design.

5.2 Future Work

In its current form the hybrid algorithm is very useful and yields high quality results. However, improvements in efficiency and extensions in capability are possible. For example, a modification that would yield further gains in efficiency is the parallelization of the GALLOP evaluations. Every GALLOP simulation in a generation is independent, and, thus, each run can be executed in parallel to significantly reduce computation time.

Genetic algorithms are also capable of optimizing combinatorial problems because of their discrete binary encoding of the design variables. In the GA-GALLOP hybrid, this capability allows for a search on the flyby sequence for LTGA trajectories. That is, the intermediate bodies that are incorporated to enable gravity assists become discrete variables so the sequence of the flyby bodies can be optimized. For example, if a LTGA trajectory from Earth to Jupiter can incorporate up to three intermediate bodies, each of the three positions in the flyby sequence can be either any gravitational body or no body at all (each position in the sequence is a design variable); although only Venus, Earth,

and Mars are practical flyby candidates. Possible trajectories are then EEVEJ, EVEMJ, EEE_J, E_ _ _ J, etc., where the launch and arrival body are included as the first and last letters respectively, and "_" indicates that a flyby does not occur at that position (e.g., EV_MJ is the sequence Earth-Venus-Mars-Jupiter). This capability accommodates many flyby sequence possibilities and allows for a less constrained, broader (and truer) global search with only three additional variables. However, the size and complexity of the optimization problem is significantly increased, requiring a large population size and inflating the computational cost.

As noted in Chapter 3, one of the most critical design considerations in the hybrid algorithm is the inheritance scheme that is employed for the design variables of the locally optimized solution from GALLOP. Both Lamarckian and Baldwinian inheritance are implemented in the GA-GALLOP hybrid algorithm, with the specification ratio of the two schemes left to the user. However, an appropriate value for this ratio is not known a priori. Empirical studies have demonstrated that employing Lamarckian inheritance for at least some members of the population improves the convergence properties of memetic algorithms, but the best ratio is likely to be problem dependent. A thorough study on the specification of this ratio for both single objective and multiobjective versions of the GA-GALLOP hybrid should be conducted. An investigation into appropriate values for the population size and elitism percentage (for the single objective implementation) is also suggested, as guidelines do not exist for memetic algorithms.

A different analysis is warranted on the required binary resolution for the design variables. The fewer the binary bits specified for the design variables, the shorter the chromosome length for every individual. A shorter chromosome length can lead to faster convergence times. Thus, guidelines for the design variable binary resolution of the design variables would be beneficial in aiding the user setup, and allow for efficient convergence. Note, however, it is critical that the resolution is sufficiently fine to encompass all possible trajectories.

In the single objective implementation of the GA-GALLOP hybrid algorithm, it may useful to incorporate a mechanism that encourages niche formation [39]. By sustaining several different types, or species, of trajectories throughout the evolution, diversity of

the population is maintained. Niching operators, such as the strategies that employ a sharing function [39], also afford genetic algorithms the capability to simultaneously optimize several high-performing solutions with different characteristics instead of only a single global optimum. In the GA-GALLOP hybrid algorithm, incorporating a niching operator may be beneficial because the top-performing trajectory in terms of final mass may also possess undesirable trajectory characteristics such as passing too close to the Sun or a short launch period. Thus, schemes for maintaining subpopulations of different trajectories during the evolution should be investigated.

In the multiobjective implementation of the hybrid algorithm, it would also be valuable to test the combination of GALLOP with multiobjective evolutionary algorithms other than the NSGA-II. Other effective multiobjective evolutionary algorithms include the N-branch tournament technique [88,89], the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [90], and the Pareto Archive Evolutionary Strategy (PAES) [91]. These different multiobjective optimization strategies, combined with GALLOP, may be more effective in developing a representation of the Pareto front or more efficient, requiring fewer generations to establish the curve of optimal solutions in terms of final mass and time-of-flight. Often the most appropriate technique is problem dependent; thus, experimenting with several different strategies and offering different options within the structure of the hybrid algorithm would be beneficial.

The characteristics of the genetic algorithm render it particularly well-suited as the global component of this hybrid algorithm. However, other global optimization methods may also be appropriate ‘wrappers’, and could possibly lead to improvements in efficiency. Specifically, differential evolution and particle swarm optimization are good candidates for incorporation as the global component, since both techniques have been successfully implemented for the global optimization of impulsive trajectories, and possess beneficial characteristics. Namely, both approaches are population-based, which appears to be a necessary distinction for a global exploration of the complex, expansive design space of low-thrust, gravity assist trajectories.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Rayman, M. D., Varghese P., Lehman D. H., and Livesay L. L., "Results from the Deep Space 1 Technology Validation Mission," *Acta Astronautica*, Vol. 47, No. 2, 2000, pp. 475-487.

[2] Rayman, M. D., Fraschetti, T. C., Raymond, C. A., and Russll, C. T., "Preparing for the Dawn Mission to Vesta and Ceres," Paper IAC-05-A3.5.B.01, 56th International Astronautical Congress, Fukuoka, Japan, October 17-21, 2005.

[3] Kawaguchi, J., Fujiwara, A., and Uesugi, T. K., "The Ion Engines Cruise Operation and the Earth Swingby of 'Hayabusa' (MUSES-C)," Paper IAC-04-Q.5.02, 55th International Astronautical Congress, Vancouver, Canada, October 4-8, 2004.

[4] Lawden, D. F., "Perturbation Maneuvers," *Journal of the British Interplanetary Society*, Vol. 13, No. 6, November 1954, pp. 329-334.

[5] Crocco, G. A., "One-Year Exploration Trip Earth-Mars-Venus-Earth," *Proceedings of the VIIth International Astronautical Congress*, Associazione Italiana Razzi, Rome, Italy, 1956, pp. 201-252.

[6] Breakwell, J. V., Gillespie, R. W., and Ross, S., "Researches in Interplanetary Transfers," *American Rocket Society Journal*, Vol. 31, No. 2, 1961, pp. 201-208.

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

[8] Debban, T. J., McConaghy, T. T., and Longuski, J. M., "Design and Optimization of Low-Thrust Gravity-Assist Trajectories to Selected Planets," Paper AIAA 2002-4729, AIAA/AAS Astrodynamics Specialist Conference, Monterey, California, August 5-8, 2002.

[9] D. Lawden, *Optimal Trajectories For Space Navigation*, Butterworths Publishers, London, 1963.

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

[11] Sauer, C. G., Jr., "Solar Electric Performance for Medlite and Delta Class Planetary Missions," Paper AAS 97-726, AAS/AIAA Astrodynamics Specialist Conference, Sun Valley, Idaho, August 4-7, 1997.

[12] Russell, R. P., "Primer Vector Theory Applied to Global Low-Thrust Trade Studies," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 460-472.

[13] Bryson, A., and Ho, Y., *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969.

[14] Hartmann, J. W., "Low-Thrust Trajectory Optimization Using Stochastic Optimization Methods," M.S. Thesis, Aerospace Engineering Department, University of Illinois at Urbana-Champaign, Urbana-Champaign, Illinois, 1996.

[15] Coverstone-Carroll, V., and Williams, S. N., "Optimal Low Thrust Trajectories Using Differential Inclusion Concepts," *Journal of the Astronautical Sciences*, Vol. 42, No. 4, October-December 1994, pp. 379-393.

[16] Seywald, H., "Trajectory Optimization Based on Differential Inclusion," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 3, 1994, pp.480-487.

[17] Tang, S., and Conway, B. A., "Optimization of Low-Thrust Interplanetary Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 18, N0. 3, May-June 1995, pp. 599-604.

[18] Herman, A. L., and Conway, B. A., "Direct Optimization Using Collocation Based on High-Order-Gauss-Lobatto Quadratic Rules," *Journal of Guidance, Control, and* Dynamics, Vol. 17, No. 3, 1994, pp. 480-487.

[19] Betts, J. T., "Optimal Interplanetary Orbit Transfers by Direct Transcription," *Journal of the Astronautical Sciences*, Vol. 42, No. 3, July-September 1994, pp. 247-268.

[20] Whiffen, G., and Sims, J., "Application of the SDC Optimal Control Algorithm to Low-Thrust Escape and Capture Trajectory Optimization," Paper AAS 02-208, AAS/AIAA Space Flight Mechanics Meeting, San Antonio, Texas, January 27-30, 2002.

[21] Kluever, C. A., "Optimal Low-thrust Interplanetary Trajectories by Direct Method Techniques," *Journal of the Astronautical Sciences*, Vol. 45, No. 3, July-September 1997, pp. 247-262.

[22] Gao, Y., and Kluever, C., "Low-Thrust Interplanetary Orbit Transfers Using Hybrid Trajectory Optimization Method with Multiple Shooting," Paper AIAA 2004 4088, AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Providence, Rhode Island, August 16-19, 2004.

[23] Howell, K., and Ozimek, M., "Low-Thrust Transfers in the Earth-Moon System Including Applications to Libration Point Orbits," Paper AAS 07-343, AAS/AIAA Astrodynamics Specialists Conference, Mackinac Island, Michigan, August 2007.

[24] Petropoulos, A. E., "A Shape-Based Approach to Automated, Low-Thrust, Gravity-Assist Trajectory Design," Ph.D. Dissertation, School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana, 2001.

[25] Petropoulos, A. E., and Longuski, J. M., "Shape-Based Algorithm for Automated Design of Low-Thrust, Gravity-Assist Trajectories," Journal Spacecraft and Rockets, Vol. 41, No. 5, September-October 2004, pp. 787-796.

[26] Izzo D., "Lambert's Problem for Exponential Sinusoids," *Journal of Guidance Control and Dynamics*, Vol. 29, No. 5, September 2006, pp. 1242-1245.

[27] De Pascale, P., and Vasile, M., "Preliminary Design of Low Thrust Multiple Gravity-Assist Trajectories," *Journal of Spacecrafts and Rockets*, Vol. 43, No. 5, September-October, 2006, pp. 1065-1076.

[28] Patel, P., Scheeres, D. J., and Zurbuchen, T., "A Shape Based Approach to Spacecraft Trajectories: Analysis and Optimization," Paper AAS 05-130, AAS/AIAA Space Flight Mechanics Meeting, Copper Mountain, Colorado, January 23-27, 2005.

[29] Fogel, D. B., *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, New Jersey, 1995.

[30] Dachwald, B., "Evolutionary Neurocontrol: A Smart Method for Global Optimization of Low-Thrust Trajectories," Paper AIAA 2004-5405, AIAA/AAS Astrodynamics Specialist Conference, Providence, Rhode Island, August 16-19, 2004.

[31] Carnelli, I., Dachwald, B., Vasile, M., Seboldt, W., and Finzi, A. E., "Low-Thrust Gravity Assist Trajectory Optimization Using Evolutionary Neurocontrollers," Paper AAS 05-374, AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, August 7-11, 2005.

[32] Hartmann, J. W., Coverstone-Carroll, V., and Williams, S. N., "Optimal Interplanetary Spacecraft Trajectories via a Pareto Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 46, No. 3, 1998, pp. 267–282.

[33] Wuerl, A., Crain, T., and Braden, E., "Genetic Algorithm and Calculus of Variations-Based Trajectory Optimization Technique," *Journal of Spacecraft and Rockets*, Vol. 40, No. 6, November-December, 2003, pp. 882-888.

[34] Woo, B., Coverstone, V. L., and Cupples, M., "Low-Thrust Trajectory Optimization Procedure for Gravity-Assist, Outer-Planet Missions," *Journal of Spacecraft and Rockets*, Vol. 43, No. 1, January-February 2006, pp. 121-129.

[35] Sentinella, M. R., and Casalino, L., "Genetic Algorithm and Indirect Method Coupling for Low-Thrust Trajectory Optimization," Paper AIAA 2006-4468, 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, Sacramento, California, July 9-12, 2006.

[36] Sauer, C. G., Jr., "Optimization of Multiple Target Electric Propulsion Trajectories," AIAA Paper 73-205, AIAA 11th Aerospace Sciences Meeting, Washington, D.C., January 10-12, 1973.

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

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

[39] Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, Massachusetts, 1989.

[40] Bäck, T., *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Oxford, 1996.

[41] Vanderplaats, G., *Numerical Optimization Techniques for Engineering Design*, 3rd ed., Vanderplaats Research & Development, Inc., Colorado Springs, Colorado, 2001.

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

[43] Goldberg, D. E., and Deb, K., "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *Foundations of Genetic Algorithms*, edited by G. Rawlins, Morgan Kaufmann Publishers, San Mateo, California, 1991, pp. 69-93.

[44] Syswerda, G., "Uniform Crossover in Genetic Algorithms," *Proceedings of the 3rd International Conference on Genetic Algorithms*, George Mason University, Fairfax, Virginia, 1989, pp. 2-9.

[45] Michalewicz, Z., "Genetic Algorithms, Numerical Optimization, and Constraints," *Proceedings of the 6th International Conference on Genetic Algorithms*, edited by L.J. Eshelman, Morgan Kaufmann Publishers, San Mateo, California, July 1995, pp. 151-158.

[46] Coello, C. A., "Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art," *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, No. 11-12, January 4, 2002, pp. 1245-1287.

[47] Deb, K., "An Efficient Constraint Handling Method for Genetic Algorithms," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, No. 2-4, June 9, 2000, pp. 311-338.

[48] Runarsson, T. P., and Yao, X., "Stochastic Ranking for Constrained Evolutionary Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, 2000, pp. 284-294.

[49] Crossley, W. A., and Cook, A. M., "Survival of Infeasible Designs during Constrained Genetic Algorithm Optimization," Paper AIAA 99-0109, 37th AIAA Aerospace Sciences Meeting & Exhibit, Reno, Nevada, January 11-14, 1999.

[50] Crossley, W. A., and Williams, E. A., "A Study of Adaptive Penalty Functions for Constrained Genetic Algorithm-Based Optimization," Paper AIAA 97-0083, 35th AIAA Aerospace Sciences Meeting & Exhibit, Reno, Nevada, January 6-9, 1997.

[51] Joines, J. A., and Houck, C. R., "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's," *Proceedings of the First International Conference on Evolutionary Computation*, Orlando, Florida, 1994, pp. 579-584.

[52] Nankani, K., Crossley, W. A., and Raymer, D. P., "Comparison of Bit-String Affinity and Consecutive Generation Stopping Criteria for Genetic Algorithms", Paper AIAA 2004-0449, 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 5-8, 2004.

[53] Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Inc., New York, New York, 2001.

[54] Zadeh, L., "Optimality and non-scalar-valued performance criteria," *IEEE Transactions on Automatic Control*, Vol. 8, No. 1, 1963, pp. 59–60.

[55] Pareto, V. *Manuale di economica politica, societa editrice libraria*, Milano, Italy, 1906; translated into English by A. Schwier as *Manual of Political Economy*, MacMillan Press, New York, 1971.

[56] Srinivas, N., and Deb, K., "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, Vol. 2, No. 3, 1994, pp. 221-248.

[57] Zitzler, E., Deb, K., and Thiele, L., "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary. Computation,* Vol. 8, No. 2, June 2000, pp. 173-195.

[58] Coello, C. A., Lamont, G. B., and Veldhuizen, D. A., *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*, 2nd ed., Springer-Verlag, New York, Inc., 2006.

[59] Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T., "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, April 2002, pp. 182-197, 2002.

[60] Williams, E. A., and Crossley, W. A., "Empirically-Derived Population Size and Mutation Rate Guidelines for a Genetic Algorithm with Uniform Crossover," *Soft Computing in Engineering Design and Manufacturing*, edited by P. K. Chawdry, R. Roy, and R. K. Pant, Springer-Verlag, New York, 1998, pp. 163-172.

[61] Schaffer, J. D., "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," *Proceedings of the 1st international Conference on Genetic Algorithms*, edited by J. J. Grefenstette, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1985, pp. 93-100.

[62] Kursawe, F., "A Variant of Evolution Strategies for Vector Optimization," *Proceedings of the 1st Workshop on Parallel Problem Solving From Nature*, edited by H. Schwefel and R. Männer, Lecture Notes In Computer Science, Vol. 496. Springer-Verlag, London, 1990, pp. 193-197.

[63] Sims, J. A., and Flanagan, S. N., "Preliminary Design of Low-Thrust Interplanetary Missions," Paper AAS 99-338, AAS/AIAA Astrodynamics Specialist Conference, Girdwood, Alaska, August 16-18, 1999.

[64] McConaghy, T. T., "Design and Optimization of Interplanetary Spacecraft Trajectories," Ph.D. Dissertation, School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana, 2004.

[65] McConaghy, T. T., Debban, T. J., Petropoulos, A. E., and Longuski, J. M., "An Approach to Design and Optimization of Low-Thrust Trajectories with Gravity Assists," Paper AAS 01-468, AAS/AIAA Astrodynamics Specialist Conference, Quebec City, QC, Canada, July-August 2001.

[66] Sims, J. A., Finlayson, P. A., Rinderle, Edward, A., Vavrina, M. A., and Kowalkowski, T. D., "Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design," Paper AIAA-2006-6746, AIAA/AAS Astrodynamics Specialist Conference, Keystone, Colorado, August 21-24, 2006.

[67] McConaghy, T. T., Debban, T. J., Petropoulos, A. E., and Longuski, J. M., "Design and Optimization of Low-Thrust Trajectories with Gravity Assists," *Journal of Spacecraft and Rockets*, Vol. 40, No.3, May-June 2003, pp. 380-387.

[68] Parcher, D. W., and Sims, J. A., "Gravity-Assist Trajectories to Jupiter Using Nuclear Electric Propulsion," Paper AAS 05-398, AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, California, August 7-11, 2005.

[69] Kowalkowski, T. D., Kangas, J. A., and Parcher, D. W., "Jupiter Icy Moons Orbiter Interplanetary Injection Period Analysis," Paper AAS 06-187, AAS/AIAA Space Flight Mechanics Meeting, Tampa, Florida, January 22-26, 2006.

[70] McConaghy, T. T., "GALLOP Version 4.5 User's Guide," School of Aeronautics and Astronautics, Purdue University, West Lafayette, Indiana, January 7, 2004.

[71] Byrnes, D. V., and Bright, L. E., "Design of High-Accuracy Multiple Flyby Trajectories Using Constrained Optimization," Paper AAS 95-307, AAS/AIAA Astrodynamics Specialist Conference, Halifax, Nova Scotia, Canada, August 14-17, 1995.

[72] Tsiolkovsky, K. E., "Exploration of the Universe with Reaction Machines," *Science Review*, No. 5, 1903, p. 31 (in Russian).

[73] Gill, P. E., "User's Guide for SNOPT Version 7: A FORTRAN Package for Large-Scale Nonlinear Programming," University of California, San Diego, December 4, 2004.

[74] Moscato, P., "On Evolution, Search, Optimization, Genetic Algorithms, and Martial Arts: Towards Memetic Algorithms," Caltech Concurrent Computation Program, Report 826, California Institute of Technology, Pasadena, California, 1989.

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

[76] Goldberg, D. E., and S. Voessner, S., "Optimizing global-local search hybrids," *Proceedings of Genetic and Evolutionary Computation Conference 99* (GECCO-99), 1999, pp. 220-228.

[77] Houck, C., Joines, J., Kay, M., and Wilson, J., "Empirical Investigation of the Benefits of Partial Lamarckianism," *Evolutionary Computation*, Vol. 5, No. 1, 1996, pp. 31-60.

[78] Bersini, H., and Renders, B., "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways," *Proceedings of the First IEEE Conference on Evolutionary Computation*, 1994, pp. 312–317.

[79] Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[80] Whitley L. D., Gordon V. S., and Mathias K. E., "Lamarckian Evolution, the Baldwin Effect, and Function Optimization," *Proceedings of the International Conference on Evolutionary Computation. the Third Conference on Parallel Problem Solving From Nature: Parallel Problem Solving From Nature*, edited by Y. Davidor, H. Schwefel, and R. Manner, Springer-Verlag, London, 1994, pp. 6-15.

[81] Hart, W. E., Krasnogor, N., and Smith, J. E., "Memetic Evolutionary Algorithms," *Recent Advances in Memetic Algorithms*, edited by E. Hart, N. Krasnogor, and J.E. Smith, Springer-Verlag, Berlin, 2004, pp. 3-27.

[82] Mason, J. C., and Handscomb, D. C., *Chebyshev Polynomials,* Chapman & Hall/CRC, Boca Raton, Florida, 2002.

[83] Yam, C. H., and Longuski, J. M., "Reduced Parameterization for Optimization of Low-Thrust Gravity-Assist Trajectories: Case Studies," Paper AIAA 2006-6744, AIAA/AAS Astrodynamics Specialist Conference, Keystone, Colorado, August 21-24, 2006.

[84] Williams, S. N., and Coverstone-Carroll, V., "Mars Missions Using Solar Electric Propulsion," *Journal of Spacecraft & Rockets*, Vol. 37, No. 1, 2000, pp. 71-77.

[85] Yam, C. H., McConaghy, T. T., Chen, K. J., and Longuski, J. M., "Design of Low-Thrust Gravity-Assist Trajectories to the Outer Planets," Paper IAC-04-A.6.02, 55th International Astronautical Congress, Vancouver, Canada, October 4–8, 2004.

[86] Yam, C. H., McConaghy, T. T., Chen, K. J., and Longuski, J. M., "Preliminary Design of Nuclear Electric Propulsion Missions to the Outer Planets," Paper AIAA-2004-5393, AAS/AIAA Astrodynamics Specialist Conference, Providence, Rhode Island, August 16-19, 2004.

[87] Petropoulos, A. E., Kowalkowski, T. D., Vavrina, M. A., Parcher, D. W., Finlayson, P. A., Whiffen, G. J., and Sims, J. A., "1st ACT Global Trajectory Optimisation Competition: Results Found at the Jet Propulsion Laboratory," *Acta Astronautica*, Vol. 61, Issue 9, November 2007, pp. 806-815.

[88]Crossley, W. A., Cook, A. M., Fanjoy, D. W., "Using the Two-Branch Tournament Genetic Algorithm for Multiobjective Design," *AIAA Journal*, Vol. 37, No. 2, November 1999, pp. 262-267.

[89] Martin, E. T., Hassan, R. A., Crossley, W. A., "Generalization of the Two-Branch Tournament for N-Objective Optimization," Paper AIAA 2002-5430, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, September 4-6, 2002.

[90] Zitzler, E., Laumanns, M., and Thiele, L., "SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm," Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 2001.

[91] Knowles, J. D., Corne, D. W., "The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation," *Proceedings of the 1999 Congress on Evolutionary Computation*, Vol. 1, 1999, pp. 98-105.