Copyright

by

Gregory Phillip Johnson

2014

The Dissertation Committee for Gregory Phillip Johnson
certifies that this is the approved version of the following dissertation:

A Tabu Search Methodology for Spacecraft Tour Trajectory Optimization

Committee:

Cesar A. Ocampo, Supervisor

Wallace T. Fowler

David P. Morton

Ryan P. Russell

Juan S. Senent

A Tabu Search Methodology for Spacecraft Tour Trajectory Optimization

by

Gregory Phillip Johnson, B.S.AS.E., M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2014

To my wife Angie, and to my parents Don and Sharon.

Acknowledgments

First and foremost, I want to thank my advisor, Dr. Cesar Ocampo. His guidance and unending support have been invaluable throughout my studies. His enthusiasm was inspiring, and he gave me the freedom to explore many different topics. I am grateful to my committee members: Dr. Wallace Fowler, Dr. David Morton, Dr. Ryan Russell and Dr. Juan Senent, for all of their support and valuable feedback. I would also like to thank Dr. J. Wesley Barnes for introducing me to tabu search and for the discussions along the way.

I have been lucky to develop many long-lasting friendships during my time at the university. I wish to thank Paul Bauman and Victor Calo for helping me to always stay positive, and in particular for their feedback during the dissertation writing process. I would also like to thank Sebastian Munoz and Divya Thakur; it was a pleasure going through the undergraduate and graduate programs with them. I also thank my research group, Mark Jesick, Ricardo Restrepo, Noble Hatten, Drew Jones and Nick Bradley, for the many discussions and the exchange of ideas.

I owe a huge debt of gratitude to my colleagues at the Texas Advanced Computing Center, especially to Kelly Gaither, Paul Navratil and Jay Boisseau, for their support during my graduate studies.

To conclude, and most importantly, I would like to thank my wife Angie and my parents Don and Sharon. None of this would have been possible without their unfailing support and patience.

A Tabu Search Methodology for Spacecraft Tour Trajectory Optimization

Publication No. ____________________

Gregory Phillip Johnson, Ph.D.
The University of Texas at Austin, 2014

Supervisor: Cesar A. Ocampo

A spacecraft tour trajectory is a trajectory in which a spacecraft visits a number of objects in sequence. The target objects may consist of satellites, moons, planets or any other body in orbit, and the spacecraft may visit these in a variety of ways, for example flying by or rendezvousing with them. The key characteristic is the target object sequence which can be represented as a discrete set of decisions that must be made along the trajectory. When this sequence is free to be chosen, the result is a hybrid discrete-continuous optimization problem that combines the challenges of discrete and combinatorial optimization with continuous optimization. The problem can be viewed as a generalization of the traveling salesman problem;

such problems are NP-hard and their computational complexity grows exponentially with the problem size. The focus of this dissertation is the development of a novel methodology for the solution of spacecraft tour trajectory optimization problems.

A general model for spacecraft tour trajectories is first developed which defines the parameterization and decision variables for use in the rest of the work. A global search methodology based on the tabu search metaheuristic is then developed. The tabu search approach is extended to operate on a tree-based solution representation and neighborhood structure, which is shown to be especially efficient for problems with expensive solution evaluations. Concepts of tabu search including recency-based tabu memory and strategic intensification and diversification are then applied to ensure a diverse exploration of the search space. The result is an automated, adaptive and efficient search algorithm for spacecraft tour trajectory optimization problems. The algorithm is deterministic, and results in a diverse population of feasible solutions upon termination. A novel numerical search space pruning approach is then developed, based on computing upper bounds to the reachable domain of the spacecraft, to accelerate the search. Finally, the overall methodology is applied to the fourth annual Global Trajectory Optimization Competition (GTOC4), resulting in previously unknown solutions to the problem, including one exceeding the best known in the literature.

Contents

Acknowledgments **v**

Abstract **vi**

List of Tables **xii**

List of Figures **xv**

List of Algorithms **xxiii**

Chapter 1 Introduction **1**

- 1.1 Spacecraft Tour Trajectory Definition 1
  - 1.1.1 Optimization Problem 2
  - 1.1.2 Relation to Traveling Salesman Problem 4
- 1.2 Motivation 5
- 1.3 Related Work 7
- 1.4 Dissertation Organization and Contributions 10

Chapter 2 Tour Trajectory Modeling **12**

- 2.1 General Model 12
  - 2.1.1 Initial Conditions 13
  - 2.1.2 Target Objects 13

2.1.3 Trajectory Segments 14
2.1.4 Tour Trajectory 15
2.2 Spacecraft Tour Trajectories 18
2.2.1 Spacecraft Dynamics 18
2.2.2 Target Object Dynamics 20
2.2.3 Finite Burn Trajectory Segments 20
2.2.4 Impulsive Trajectory Segments 24
2.2.5 Impulsive Maneuver to Finite Burn Maneuver Conversion . . 27
2.2.6 Augmented Impulsive Tour Model 36

Chapter 3 Global Search Methodology **39**
3.1 Solution Representation 40
3.1.1 Properties of Tour Trajectories 41
3.1.2 Tree-Based Solution Representation 45
3.2 Neighborhoods 53
3.2.1 Neighborhoods for the Tree Solution Representation 54
3.2.2 Restricted Best-First Neighborhood 60
3.3 Objectives 62
3.3.1 Guiding Objective 63
3.3.2 Budget Penalty Terms 67
3.4 Solution Construction (Node Expansion) 70
3.5 Tabu Search 73
3.5.1 Recency-based Tabu Memory 75
3.5.2 Strategic Intensification and Diversification 81
3.6 Algorithm Summary 87

Chapter 4 Search Space Pruning **92**
4.1 Brute-force Approach 92

4.2 Trajectory Envelopes 95
4.2.1 Bounding Boxes 100
4.2.2 Summary 106
4.3 Performance 107

Chapter 5 Application to Fourth Global Trajectory Optimization Competition 113

5.1 Problem Definition 115
5.1.1 GTOC4 Augmented Impulsive Tour Model 117
5.2 Best Known Solutions 120
5.3 Results 120
5.3.1 Base Case 122
5.3.2 Finite Burn Constraints 129
5.3.3 Search Space Pruning 132
5.3.4 Dynamic Neighborhood Selection 136
5.3.5 Reduced Spacecraft Performance 142
5.3.6 Comparison to GTOC4 Winning Solution 152
5.3.7 Low-thrust Finite-burn Conversion 159
5.4 Summary 164

Chapter 6 Conclusions 169

6.1 Dissertation Summary 169
6.2 General Conclusions 171
6.3 Future Work 172

Appendices 174

Appendix A Software Implementation 174

A.1 Tree-Based Solution Representation 174

Appendix B Set of GTOC4 Asteroids **180**

Bibliography **208**

Vita **217**

List of Tables

2.1 Decision variables for the general tour trajectory model. 16

2.2 Spacecraft parameters and descriptions. 18

3.1 Summary of components and parameters for the tabu search algorithm. 90

4.1 Sampling parameters for trajectory segments in the augmented impulsive tour model. 93

4.2 Parameters for the trajectory envelope search space pruning procedure. 107

4.3 Search space pruning parameters and trajectory segment sampling parameters for performance analysis of the space pruning procedure. 108

4.4 Results of the search space pruning approach applied to the GTOC4 problem over varying values of ΔV_{max}. 110

5.1 Summary of past Global Trajectory Optimization Competition (GTOC) problems and their mission sequences to be optimized [1]. 114

5.2 GTOC4 mission and spacecraft parameters [12]. 115

5.3 The sun's gravitational parameter and Earth orbital elements for GTOC4 problem in J2000 heliocentric ecliptic reference frame [12]. . 116

5.4 Final results of the fourth Global Trajectory Optimization Competition (GTOC4) [11]. 121

5.5 For each case, a collection of 1024 runs are generated over a range of launch epochs. Each run executes for 2 hours, requiring 2048 compute hours in total. 121
5.6 Tabu search algorithm parameters for the base case. 123
5.7 Base case: the minimum, median and maximum best solutions found over the 1024 runs. 124
5.8 Comparison of best solutions found in the GTOC4 competition results and the base case [11]. 125
5.9 Base case: run #409/1024. The tour itinerary is shown for a $J = -45$ solution rendezvousing with asteroid 2006BZ147. 127
5.10 Spacecraft parameters for base case and reduced spacecraft performance cases. 142
5.11 Summary of best solutions found for base case and reduced spacecraft performance cases. 143
5.12 Reduced performance case A: run #622/1024. The tour itinerary is shown for a $J = -45$ solution rendezvousing with asteroid 2006QQ56. 146
5.13 Reduced spacecraft performance case B: run #402/1024. The tour itinerary is shown for a $J = -43$ solution rendezvousing with asteroid 2006UB17. 151
5.14 Increased spacecraft performance case: The tour itinerary is shown for a $J = -45$ solution rendezvousing with asteroid 2000SZ162. . . . 156
5.15 Low-thrust finite burn trajectory corresponding to the $J = -45$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.16 and Table 5.12. The tour itinerary is shown. 163
5.16 Low-thrust finite burn trajectory corresponding to the $J = -43$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.18 and Table 5.13. The tour itinerary is shown. 167

A.1 Tree traversal methods implemented in the *TreeNode* class. 176

List of Figures

2.1 A single trajectory segment and its associated initial conditions and decision variables. $\mathbf{z}_{i-1}(t_{i-1})$ represents the state at the initial time. s_i denotes the target object, and $\mathbf{y}_i$ gives the continuous decision variables of the segment including the segment duration Δt_i. 15

2.2 A tour trajectory and its associated decision variables. $\mathbf{z}_0$ represents the initial state at time t_0. $s_1 \ldots s_{n_s} \in O$ denote the sequence of objects visited in the tour, and $\mathbf{y}_1 \ldots \mathbf{y}_{n_s}$ give the continuous decision variables of each segment. The tour consists of n_s trajectory segments visiting n_s target objects. 16

2.3 A finite burn trajectory segment. An initial maneuver causes the spacecraft to intercept the target object, and an optional final maneuver causes a rendezvous. 21

2.4 An impulsive trajectory segment. An initial impulse $\Delta\mathbf{V}_{int}$ maneuvers the spacecraft to intercept the target object, and an optional final impulse $\Delta\mathbf{V}_{ren}$ maneuvers the spacecraft to rendezvous with the target. 25

2.5 The impulsive trajectory segment shown in Figure 2.4 with all spacecraft states and discontinuities shown. 26

2.6 A valid finite burn representation of an impulsive $\Delta\mathbf{V}$ maneuver The finite burn begins at time t_{fb_0}, thrusts over a period of time and matches position and velocity with the post-impulse trajectory at time t_{fb_f} [50]. 28

2.7 An impulsive trajectory segment with estimates for finite burn maneuvers replacing the impulses. 30

2.8 The finite burn characteristic velocity and duration ratios are shown for the case where gravity losses are considered and for the ideal case. 50 consecutive maneuvers are made for an impulsive ΔV of 1 km/s. The spacecraft parameters correspond to the GTOC4 problem described in Chapter 5. 35

3.1 k tour trajectories are shown with final states $\mathbf{z}_{n_s}^1 \dots \mathbf{z}_{n_s}^k$. Each trajectory shares the previous $n_s - 1$ segments in common. The state $\mathbf{z}_{n_s-1}$ can be used in the computation of the k final segments. 43

3.2 A group of trajectories. Colors identify different objects. The final bright green object is visited with three different segments corresponding to different segment decision variables $\mathbf{y}$. 46

3.3 A detailed view of the tree representation of Figure 3.2 is shown. Children of nodes are numbered in ascending order starting at 0. Paths can be defined with ordered lists of node numbers starting at the root of the tree. States and decision variables are annotated for two paths, corresponding to the superscripts 1 and 2. 47

3.4 All possible tour sequences shown in the tree solution representation for $N_o = 5$ candidate objects and sequence lengths of $n_s = 5$. 49

3.5 Number of tours and tree solution representation speedup for total enumeration of $N_o = 100$ candidate objects and segment discretizations of $K = 1, 2, 4, 8$. 51

3.6 An annotated tree is shown for an incumbent solution $\mathbf{x}$. The root of the tree $\mathbf{x}_{root}$, children nodes $\mathcal{C}(\mathbf{x})$ and parent nodes $\mathcal{P}^k(\mathbf{x})$ are labeled. 55

3.7 The neighborhood definition $\mathcal{N}(\mathbf{x}) = \mathcal{C}(\mathbf{x})$ leads to a depth-first search. At each iteration the incumbent solution moves deeper in the tree. 56

3.8 The neighborhood corresponds to a breadth-first search. The search explores all nodes at at the highest level of the tree before progressing deeper. 58

3.9 A group of tour trajectories represented in the tree solution representation. The best-first neighborhood is highlighted. 59

3.10 A group of tour trajectories represented in the tree solution representation. The restricted best-first neighborhoods corresponding to different h values are shown. 61

3.11 The guiding objective contains the objective defined in the problem statement evaluated on the partial trajectory $J(\mathbf{x})$, and a heuristic term based on the parameter $\frac{dJ}{dt}^{\star}$ estimating contributions over the remaining mission time. 64

3.12 The mass and ΔV time histories are shown for maximum continuous thrust until fuel exhaustion and the budgeted amount. The regions where penalties are added to the objective are highlighted. The infeasible regions for a finite burn model and for the final state constraints are shown. The spacecraft parameters correspond to the GTOC4 problem [12]. 69

3.13 For the next target object s, a discretized grid of possible new children solutions over the range of allowed τ and Δt values. The set of solutions found to be feasible after evaluation is highlighted. 72

3.14 The tabu list $\mathcal{T}$ is an array of solution attributes that are prohibited in the search. Its length is the tabu tenure $N_{\mathcal{T}}$ and determines how many iterations attributes are considered tabu. As attributes of new incumbent solutions $A(\mathbf{x}_i)$ are added to the list, the oldest tabu attributes are forgotten. 78

3.15 An example search history. The incumbent solution lengths are shown along with the stall condition evaluated for $N_{stall} = 25$ and $\underline{R} = 5$. . 83

4.1 The values for ΔV_{max} are shown over varying spacecraft initial mass and values of Δt_{max}. The spacecraft parameters correspond to the fourth annual global trajectory optimization competition (GTOC4) summarized in Table 5.2. 97

4.2 Discretization of all possible impulsive maneuvers for a magnitude of ΔV_{max}. The maneuver direction is discretized over the spherical angles α and β for a total of $K_{\Delta V} = K_\alpha K_\beta$ discretizations. 99

4.3 The trajectories forming a trajectory envelope are shown for $\Delta V_{max} =$ 1 km/s and $\Delta t_{max} = 0.55$ years. The spherical angles α and β are each discretized over a 15 degree spacing. 100

4.4 An axis-aligned bounding box bounds a volume in the spatial region of $([x_{min}, x_{max}], [y_{min}, y_{max}], [z_{min}, z_{max}])$. We additionally constrain the bounding box with a time range $[t_{min}, t_{max}]$. 102

4.5 $N_{BB} = 10$ bounding boxes are shown for the trajectory envelope of Figure 4.3 ($\Delta V_{max} = 1$ km/s). 105

4.6 The target object bounding box intersections are shown for the case in Figure 4.5. 105

4.7 The bounding boxes associated with trajectory envelopes from the results in Table 4.4. 111

5.1 The set of 1436 asteroids for the GTOC4 problem. The highlighted regions show combinations of semimajor axis and eccentricity for Atiras, Atens, Apollos and Amors near-Earth asteroids [45]. 116
5.2 Base case: Best partial and complete solutions found for 1024 runs over launch dates from 2015 to 2025. 124
5.3 Base case: run #409/1024. The objective history for the incumbent solution and best found partial and complete solutions are shown. . 126
5.4 Base case: run #409/1024. The trajectory is shown for a $J = -45$ solution rendezvousing with asteroid 2006BZ147. 128
5.5 The best solutions found for the base case and the case ignoring finite burn constraints. 130
5.6 The number of trajectory families found for complete solutions in the base case and the case ignoring finite burn constraints. 131
5.7 Performance of base case (search space pruning) versus case with space pruning disabled for 1024 runs. The run time is constrained to 2 hours. A median of 10.5× more iterations are computed with space pruning enabled, with 95% of runs seeing at least a 5× improvement. 133
5.8 Results of base case (search space pruning) versus case with space pruning disabled for 1024 runs. The run time is constrained to 2 hours. The best partial and complete solutions are shown. 134
5.9 Performance of base case (search space pruning) versus case with space pruning disabled for 1024 runs. 1000 iterations are computed and the run time is unconstrained. The median runtime speedup is 11.0× with 99% of the runs seeing at least a 5× speedup. 135

5.10 The results of the base case are re-run with dynamic neighborhood selection disabled for various static values of the restricted best-first neighborhood parameter h. The best solutions found are shown for each case. 138
5.11 The best solutions found for the static neighborhood $h = \infty$ case and the dynamic neighborhood selection case. 139
5.12 The number of trajectory families found for complete solutions in the dynamic neighborhood selection case and the static neighborhood $h = \infty$ case. 140
5.13 The standard deviation of the incumbent solution length for all runs in the dynamic neighborhood selection case and the static neighborhood $h = \infty$ case. 141
5.14 Reduced spacecraft performance cases compared to base case: Best partial and complete solutions generated by 1024 runs over launch dates from 2015 to 2025. 144
5.15 Reduced spacecraft performance case A: run #622/1024. The objective history for the incumbent solution and best found partial and complete solutions are shown for (a) 2 hour run time and (b) 36 hour run time. 145
5.16 Reduced performance case A: run #622/1024. The trajectory is shown for a $J = -45$ solution rendezvousing with asteroid 2006QQ56. 147
5.17 Reduced spacecraft performance case B: run #402/1024. The objective history for the incumbent solution and best found partial and complete solutions are shown for (a) 2 hour run time and (b) 36 hour run time. 148

5.18 Reduced spacecraft performance case B: run #402/1024. The trajectory is shown for a $J = -43$ solution rendezvousing with asteroid 2006UB17. 150

5.19 Number of trajectory families found versus objective for the base case parameters and a launch epoch of 58676.40 MJD. The search is allowed to run for 12 hours. Only complete solutions are shown. . . 153

5.20 Number of trajectory families found versus objective for the increased spacecraft performance case and a launch epoch of 58676.40 MJD. The spacecraft's maximum allowed thrust T_{max} is increased by 5% from 0.135 N to 0.14175 N. The search is allowed to run for 12 hours. Only complete solutions are shown. 155

5.21 Increased spacecraft performance case: The trajectory is shown for a $J = -45$ solution rendezvousing with asteroid 2000SZ162. 157

5.22 Workflow for GTOC4 solution process. The impulsive search algorithm generates a population of impulsive trajectories. The user then selects specific impulsive trajectories, optimizes them impulsively, and passes them to the black-box finite burn conversion and optimization tool. 160

5.23 Low-thrust finite burn trajectory corresponding to the $J = -45$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.16 and Table 5.12. 161

5.24 Low-thrust finite burn trajectory corresponding to the $J = -45$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.16 and Table 5.12. The original impulsive trajectory is also shown. 162

5.25 Low-thrust finite burn trajectory corresponding to the $J = -43$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.18 and Table 5.13. 165

5.26 Low-thrust finite burn trajectory corresponding to the $J = -43$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.18 and Table 5.13. The original impulsive trajectory is also shown. 166

A.1 The software package that implements the tabu search methodology allows users to interactively run, adjust and analyze results of the search through its user interface and visualization capabilities. . . . 175

A.2 The *TreeNode* class represents nodes in the tree-based solution representation. A *TreeNode* object is shown with the parent and children nodes it references. The collection of *TreeNode* objects combine to form the search tree. 177

List of Algorithms

1 Local search algorithm. 40
2 Adaptively update guiding objective heuristic 67
3 Basic tabu search algorithm [63]. 76
4 Test if a solution $\mathbf{x}$ is tabu. 80
5 Update tabu attributes. 80
6 Intensify the search about the solution $\mathbf{x}$. 85
7 Diversify the search about the solution $\mathbf{x}$. 86
8 Escape from the region of the search space near the solution $\mathbf{x}$. . . . 86
9 Dynamically update the neighborhood based on the search performance. 88
10 Tabu search algorithm. 91
11 Expand the bounding box b given the position (x, y, z) at time t. . . 103
12 Test if the bounding box b contains the position (x, y, z) at time t. . 104
13 Example getLeaves() implementation. 178

Chapter 1

Introduction

1.1 Spacecraft Tour Trajectory Definition

A spacecraft tour trajectory can be defined as a trajectory in which a spacecraft visits a number of objects in sequence. The collection of target objects may consist of satellites, moons, planets or any other body in orbit. The spacecraft may visit these in a variety of ways, for example flying by or rendezvousing with the targets. The conditions that must be satisfied at each target are problem-specific, as are the dynamics of the trajectory and constraints on the spacecraft. The key characteristic however is the target object sequence: this is a discrete set of decisions that must be made along the trajectory. For some missions the sequence may be predetermined, while for others it may be designed to optimize mission objectives. This work focuses on the latter case. The sequence of a tour trajectory may be designed by hand; however, as mission complexity grows this quickly becomes challenging. For such problems there may be a vast number of potential sequences, and enumerating even a fraction of them is not practical. Further, for each single sequence there may exist a continuum of solutions. Developing an automated methodology for finding promising spacecraft tour trajectories, subject to problem-provided constraints and objectives, is the subject of this work.

1.1.1 Optimization Problem

We can formulate the spacecraft tour trajectory design problem as an optimization problem. The decision variables of the optimization include two principal components: the target object sequence of the trajectory, and the remaining decision variables representing initial conditions, timing and maneuvering information, for example. The sequence decision variables are discrete, belonging to a finite (but potentially very large) set of possible values. Conversely, the remaining decision variables are continuous. The resulting problem is thus a hybrid discrete-continuous optimization problem combining the challenges of discrete and combinatorial optimization with continuous optimization.

We can examine the overall problem by considering its simplifications. First, consider the target object sequence to be predetermined and fixed. This eliminates the discrete decision variables and yields a continuous optimization problem that is treatable using optimal control methods. Numerical methods for continuous optimal control problems can be broadly classified as either indirect or direct methods [13]. Indirect methods are based on analytical necessary conditions from the calculus of variations, and usually require the solution of a nonlinear multipoint boundary value problem [33]. Direct methods instead introduce a parameterization for the control variables, transcribing the continuous optimal control problem into a parameter optimization problem. The control variables are then manipulated directly to optimize the objective function, resulting in a nonlinear programming problem (NLP) [48]. The general optimal control problem is a global optimization problem; however both the indirect and direct methods described so far only find locally optimal solutions. These formulations can be used in multi-stage workflows, for example executing the optimization for a variety of starting conditions [60] or in combination with metaheuristic approaches [18, 67], to explore the global search space. Further, metaheuristic methods can be used on their own to find optimal solutions in the global search space [66, 61].

We can also simplify the problem by eliminating the continuous decision variables, either by fixing their values or disregarding the dynamics of the problem. The result is then a discrete optimization problem that only considers the target object sequence of the trajectory. If we represent the sequence with integer decision variables, the result is an integer programming problem (IP) [47]. Integer programming problems are NP-hard, and it is widely believed that no polynomial-time algorithms exist for their solution [53]. We can classify solution methods for integer programming problems as either exact or approximate. An exact method finds a provably optimal solution, while an approximate method finds good solutions with no guarantee of optimality. The commonly used exact method for solving integer programming problems is branch and bound. The algorithm systematically enumerates candidate solutions, discarding subsets of solutions based on estimates for upper and lower bounds on the objective being optimized. In the case of minimization, lower bounds are often found by solving relaxations of the original problem–for example by allowing integer decision variables to be continuously valued. Upper bounds representing worst-case best solutions are generated as integer-feasible solutions are found. Dynamic programming is another method applicable to integer programming problems that can be decomposed into simpler subproblems in a recursive manner. The method is based on Bellman's principle of optimality [10]. Both branch and bound and dynamic programming find globally optimal solutions; however the required computational effort make them impractical for many larger-scale problems. In these cases approximate methods such as metaheuristics can be used. These include evolutionary and genetic algorithms, simulated annealing, particle swarm optimization and tabu search, among others [43]. These methods generally do not find provably global optimal solutions; their purpose is instead to find good solutions in reasonable compute times. For both exact and approximate methods, space pruning approaches can be used to reduce the size of the search space and reduce the time to solution. For example, cutting plane methods use valid inequalities to prune the search space; when combined with branch and bound these form the basis for branch and cut algorithms [47]. Similarly, other space pruning procedures can be combined with approximate methods to achieve speedups [35].

When we combine both the discrete and continuous components of the optimization problem, we can form a mixed-integer nonlinear programming problem (MINLP) [22]. The focus of this work is on large-scale spacecraft tour trajectory problems, and thus we explore metaheuristic methods for their solution in this study.

1.1.2 Relation to Traveling Salesman Problem

The traveling salesman problem (TSP) is perhaps the most widely studied problem in discrete and combinatorial optimization, and can be considered as a basis for the spacecraft tour trajectory optimization problem [40]. The problem considers a salesman that departs his home city, visits each of a collection of cities, and then returns to his home city upon completion. The optimal tour minimizes the total distance traveled by the salesman. For N cities, there are a total of $(N-1)!$ possible tours starting from the home city. If the distance between cities is the same regardless of direction, then the problem is symmetric and the search space can be reduced to $\frac{(N-1)!}{2}$ solutions. For $N = 10$, there are nearly 200,000 possible solutions; for $N = 100$, there are more than 10^{155}. Total enumeration is clearly not feasible except for the smallest problem instances. In 1954, Dantzig solved a 49-city TSP, establishing a record that held for 17 years [19]. As of this writing, the largest known solved instance of the traveling salesman problem consists of 85,900 cities, and was found by the Concorde solver using cutting plane methods [5].

We can adapt the TSP to fit the spacecraft tour trajectory optimization problem by considering the salesman to be the spacecraft, the cities to be target objects, and the objective to minimize fuel consumption or mission time, for example. Further, there exist variations of the TSP which can be mapped to other components of spacecraft tour trajectory problems [30]. While the standard TSP considers visiting the entire collection of cities, the orienteering problem instead requires visiting only a subset of the cities. Each city has an associated value or prize, and the objective is to find a tour maximizing the total collected prize. Travel between cities has an associated duration, and the tour must not exceed a problem-specified total duration. The duration constraint can be mapped to either a maximum time limit or fuel limit on a spacecraft trajectory, and the prize associated with each city can be mapped to mission value per target object, for example. Another variation is the time dependent TSP, which considers the cost of travel between cities to vary with each time period. The moving target TSP further complicates the problem by assuming all cities are moving at some fixed velocity. The combination of these latter two variations could be used to approximate the dynamics and maneuvering costs of a spacecraft trajectory, for example. The close relation of the TSP to the spacecraft tour trajectory optimization problem highlights the applicability of discrete and combinatorial optimization methods to the current work.

1.2 Motivation

The main purpose of this dissertation is the development of a novel methodology for solving spacecraft tour trajectory design and optimization problems. The search space for problems of this type is sufficiently large to make exact approaches impractical. Therefore, the focus of this work is the development of an efficient algorithm that finds promising, but not provably globally optimal, solutions quickly. We base our approach on the tabu search metaheuristic, adapting and extending the method to the spacecraft tour trajectory optimization problem, and benchmark the method against known problems in the literature [27].

There are many examples of interplanetary missions that can be considered spacecraft tour trajectories. These include the Voyager 1 and 2, Cassini-Huygens, Messenger and Rosetta missions, among others. Each of these trajectories executes a sequence of gravitational flybys, and could therefore have been posed as a spacecraft tour trajectory optimization problem. However, the primary motivation for this work is problems with much larger design spaces. An example of practical importance is the Earth orbital debris problem: as the amount of debris in low Earth orbit increases, the collision risk to current and future space missions grows [37]. Even with no new satellite launches, in the absence of any mitigation strategy this population will continue to increase due to collisional cascading, a behavior known as the Kessler syndrome [38]. As a result, recent studies have emphasized the need for active debris removal (ADR) to control the orbital debris population [41, 42]. One concept for ADR is the design of missions to rendezvous with multiple debris objects for the purposes of mitigation [6, 14, 15]. This is a challenging problem, requiring the design of a trajectory that visits a subset of debris objects out of a population of thousands of potential targets. The methodology developed in this work is directly applicable to such problems.

The Global Trajectory Optimization Competition (GTOC) is another primary motivation for this study [1]. The GTOC is an international competition focusing each year on challenging global optimization problems in interplanetary trajectory design. Each of the past seven competition problems have belonged to the class of discrete-continuous optimization problems that is the focus of this work. The GTOCs have enjoyed wide participation, and the competition results serve as a useful set of benchmark solutions for testing new methodologies for spacecraft tour trajectory optimization. The fourth GTOC problem, a multiple asteroid intercept and rendezvous mission, is specifically considered in this work [12].

1.3 Related Work

Recently there has been an increase in research related to the spacecraft tour trajectory problem, including applications of both exact and approximate methods for discrete-continuous optimization. At the time of Betts's survey paper, he made the claim that trajectory optimization problems do not fall into the class of problems with discrete decision variables, and therefore there was no reason to use such methods [13]. Although this was more true at that time, this is becoming no longer the case. Here we give a brief overview of some of the relevant contributions in this area.

Alemany conducted a survey in 2007 on global optimization for low-thrust multiple asteroid tour missions [4]. At the time, there were no fully integrated methods in the literature for optimizing full tour trajectories, including the mission sequence. Instead, the sequences were predetermined using other approaches before passing them to a continuous optimization method. Alemany later developed a branch and bound approach that she combined with pruning techniques to systematically explore the asteroid sequences as well [3]. The pruning techniques were limited to rendezvous missions only, however, ignoring the possibility for flybys. Cerf later successfully applied branch and bound to a multiple space debris collection mission [15]. He points out, however, that the approach is limited to smaller problem sizes, and for large-scale problems it would be necessary to explore alternate approaches. A dynamic programming based approach was used by Grigoriev to generate the winning solution for the fourth Global Trajectory Optimization Competition (GTOC4) [29]. That method relied on a heuristic procedure to significantly prune the search space, and thus the final solution was not provably globally optimal. In general, the successful application of these exact procedures depends on either a small problem size or a significant problem-specific pruning of the search space.

There have been various approaches to addressing tour problems using metaheuristics, many of which have been hybrid approaches. Sentinella developed a hybrid evolutionary algorithm for interplanetary trajectories with multiple impulses and gravity assists [61]. The method makes use of genetic algorithms, differential evolution and particle swarm optimization in parallel . The approach finds globally good solutions, but does not vary the discrete decision variables of the problem–the mission sequences are fixed. Izzo applies differential evolution, particle swarm optimization, and genetic algorithms to multiple gravity assist trajectories in a similar way [35]. Another hybrid approach by Woo uses a genetic algorithm for global search and refines solutions with an indirect calculus of variations based method [72], while Shan does the same but instead combines particle swarm optimization with a direct method [62]. Vinko benchmarks several global optimization metaheuristics on tour problems as well [69]. In all of these cases, the metaheuristic methods are used to find globally good solutions for fixed mission sequences, not treating the discrete decision variables of the problems.

Other approaches treat the combinatorial components of the tour problem serially in a multi-stage manner. Izzo presents a three-stage method to for an asteroid grand tour problem [36]. The first stage treats the combinatorial problem, finding promising sequences based on a generalized distance metric, while the latter two stages treat the global and local optimization of the problem for the given sequence. Olympio formulates the low-thrust multiple asteroid tour problem as an optimal control problem, utilizing an indirect method for its solution [52]. It takes as input an impulsive tour with a predetermined sequence to generate the low-thrust optimal result, and can thus be used as a final stage for other methods.

A limited number of approaches have varied the mission sequences directly in the metaheuristic search procedures. Vasile combined an evolutionary algorithm with a systematic branching strategy [65]. In some of the results, the mission sequences were left free in the optimization, and promising results were found for flyby sequences to Jupiter. However, the author recommended that the combinatorial components of the problem be treated separately. Morimoto implements a basic genetic algorithm for multiple asteroid sample return missions [46]. In that work the asteroid sequence was left free (but of fixed length), and missions visiting sequences of up to three asteroids were found.

Finally, custom approaches not based on a particular metaheuristic have been used. Barbee created the series method for finding promising tours of multiple small bodies, considering both rendezvous and flyby, and applied the methodology to the fourth Global Trajectory Optimization Competition [7]. The method iteratively constructs the tour, at each step generating a population of additional trajectory segments and choosing the next segment in a greedy nearest-neighbor manner. The method therefore does not explore the global search space, but executes with a reduced computational complexity. Later, he applied the same method for designing missions to remove multiple orbital debris objects [6]. In both cases, both the sequence of target objects and continuous decision variables were allowed to vary simultaneously. However, the optimality of the series method has not been determined.

The methods surveyed span exact and approximate methods in discrete and combinatorial optimization, global optimization, as well as direct and indirect methods in continuous trajectory optimization. However, few methods treat both the mission sequence and continuous decision variables in a unified approach. The author notes however that there are undoubtedly other approaches not documented in the literature that have been developed and applied to the several Global Trajectory Optimization Competition problems [1].

1.4 Dissertation Organization and Contributions

The current dissertation describes an overall methodology for the solution of spacecraft tour trajectory optimization problems for the purposes of preliminary and conceptual mission design. The work focuses on large-scale problems, and applies many of the tenets of tabu search. The resulting method is intended to be broadly applicable to spacecraft tour trajectory optimization problems.

Chapter 2 develops a general model for tour trajectories that visit a collection of target objects in sequence. Cases are modeled where both the agent and target objects move with time according to some set of prescribed dynamics. This general model defines the parameterization and decision variables that are used in subsequent models and the development of the global search methodology. A model for spacecraft tour trajectories subject to two-body dynamics utilizing impulsive maneuvers is then developed for use in later applications. Chapter 3 develops the global search methodology, based on the tabu search metaheuristic, for finding promising solutions to tour trajectory optimization problems. It first describes a tree-based solution representation for tour trajectories, and then defines neighborhoods that operate on that representation. It then presents guiding objective functions for use in the search, and describes the use of recency-based tabu memory as well as strategic intensification and diversification. The result is an algorithm based on the tabu search metaheuristic, which is the first known application of tabu search to spacecraft trajectory optimization. Chapter 4 then develops a novel numerical method for search space pruning which can be used to accelerate the tabu search algorithm. The approach efficiently computes an upper bound to the reachable domain of the spacecraft that is used to prune the search space and reduce the number of infeasible trajectories explored during the search. Finally, Chapter 5 applies all of the components of the methodology to the fourth annual Global Trajectory Optimization Competition (GTOC4). It combines the impulsive spacecraft tour trajectory model, the tabu search algorithm and the search space pruning method. A sensitivity analysis is conducted to study the effect of each component of the algorithm. Selected solutions are converted to optimal finite burn trajectories, generating new previously unknown solutions to GTOC4. Chapter 6 then draws general conclusions and presents possibilities for future work.

Chapter 2

Tour Trajectory Modeling

This chapter develops models for tour trajectories that visit a collection of target objects in sequence. We model cases where both the agent and target objects are allowed to move with time according to some set of prescribed dynamics. These tour trajectories therefore represent complications of simpler models such as that of the traveling salesman problem. A general model is first developed; this defines the parameterization and decision variables for use in subsequent models and the global search methodology discussed in Chapter 3, "Global Search Methodology". We then develop a model for spacecraft tour trajectories considering their specific dynamics that is used for applications in Chapter 5, "Application to Fourth Global Trajectory Optimization Competition".

2.1 General Model

We develop a general model for tour trajectories that represents all of the decision variables applicable to the types of tour problems we will consider. Like the traveling salesman problem, it must first represent the sequence of target objects the agent must visit. However, now we assume that the agent and target objects move with time, and therefore may be visited at different times resulting in different costs. There are a continuum of ways with associated costs to move from one target to another, rather than the singular static costs of moving between objects in the traveling salesman problem. The tour problem thus has both discrete decision variables to represent the sequence and continuous decision variables that determine the properties of the specific path taken.

2.1.1 Initial Conditions

The tour begins with the initial conditions of the agent. This includes the time at which the tour starts, t_0, and the initial state of the agent at that time. Since the agent moves according to some dynamics, the state contains the velocity in addition to the position. Specific problems may include additional elements in the state such as the agent's mass or available fuel, for example. The components of the state are given in the state vector $\mathbf{z}$. Thus, the decision variables for the initial conditions are

$$t_0 \qquad \text{initial time} \tag{2.1}$$

$$\mathbf{z}_0(t_0) = \begin{pmatrix} \mathbf{r}(t_0) \\ \mathbf{v}(t_0) \end{pmatrix} \qquad \text{agent state at initial time} \tag{2.2}$$

The initial conditions may be free or constrained in the problem statement. For example, they may be constrained to match the state of an initial target object at t_0.

2.1.2 Target Objects

We have a collection of target objects that the agent may visit during the tour. We define these target objects with the set O,

$$O = \{O_1, O_2, O_3, \ldots, O_{N_O}\} \tag{2.3}$$

In the traveling salesman problem, these target objects represented cities with fixed positions and therefore fixed distances between them. We now allow them to move with time and assume that their position and velocity are known for all time. Their state can be computed according to known dynamics or retrieved by some other means such as an ephemeris. We define the state of a target object O_i with its position and velocity as

$$\mathbf{X}_{O_i}(t) = \begin{pmatrix} \mathbf{r}_{O_i}(t) \\ \mathbf{v}_{O_i}(t) \end{pmatrix} \tag{2.4}$$

With the addition of the velocity to the state, there are now two ways that we may visit a target object. We say that the agent intercepts the target it matches the target's position at a specified time. The agent rendezvouses with the target if it additionally matches the target's velocity. If we visit a target object O_i at time t_k, then the conditions for an intercept and rendezvous are

$$\mathbf{r}(t_k) = \mathbf{r}_{O_i}(t_k) \qquad \text{(intercept)} \tag{2.5}$$

$$\begin{pmatrix} \mathbf{r}(t_k) \\ \mathbf{v}(t_k) \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{O_i}(t_k) \\ \mathbf{v}_{O_i}(t_k) \end{pmatrix} \qquad \text{(rendezvous)} \tag{2.6}$$

2.1.3 Trajectory Segments

We define a trajectory segment for each target object in a tour sequence. A segment begins at a given time and state t_{i-1} and $\mathbf{z}_{i-1}$ and ends at a target object specified by the discrete decision variable $s_i \in O$. We also associate continuous decision variables $\mathbf{y}_i$ with the segment; at a minimum this contains the segment duration Δt_i, but can also have elements to represent maneuvers or other properties of the trajectory to s_i. We then have

$$\mathbf{y}_i = \begin{pmatrix} \Delta t_i \\ \vdots \end{pmatrix} \tag{2.7}$$

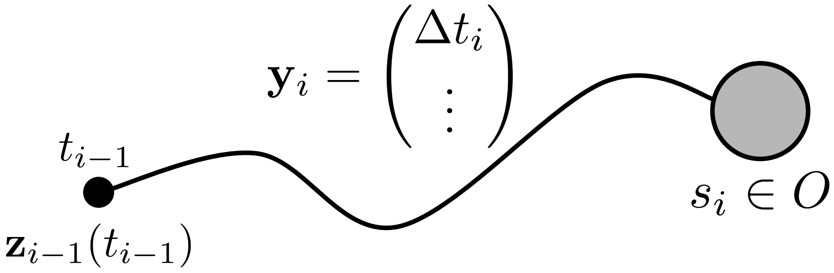

$\mathbf{y}_i = \begin{pmatrix} \Delta t_i \\ \vdots \end{pmatrix}$
t_{i-1}
$\mathbf{z}_{i-1}(t_{i-1})$
$s_i \in O$

Figure 2.1: A single trajectory segment and its associated initial conditions and decision variables. $\mathbf{z}_{i-1}(t_{i-1})$ represents the state at the initial time. s_i denotes the target object, and $\mathbf{y}_i$ gives the continuous decision variables of the segment including the segment duration Δt_i.

The time at the end of the segment t_i is found from the known initial time t_{i-1} and the segment duration Δt_i as

$$t_i = t_{i-1} + \Delta t_i \tag{2.8}$$

Given the initial conditions $\mathbf{z}_{i-1}$ at t_{i-1} and the segment decision variables s_i and $\mathbf{y}_i$, we can compute the state at the end of the segment $\mathbf{z}_i$ according to the prescribed dynamics functionally as

$$\mathbf{z}_i(t_i) = \mathbf{z}_i\left(t_{i-1}, \mathbf{z}_{i-1}, s_i, \mathbf{y}_i\right) \tag{2.9}$$

This assumes that any required maneuvers can be computed for either intercepting or rendezvousing with the target object. This is dependent on the dynamics of the specific problem; we consider specific cases for spacecraft tour trajectories in later sections. Figure 2.1 shows a single trajectory segment.

2.1.4 Tour Trajectory

We can patch multiple trajectory segments together to form a complete tour trajectory, where the final state of one segment corresponds to the initial conditions of

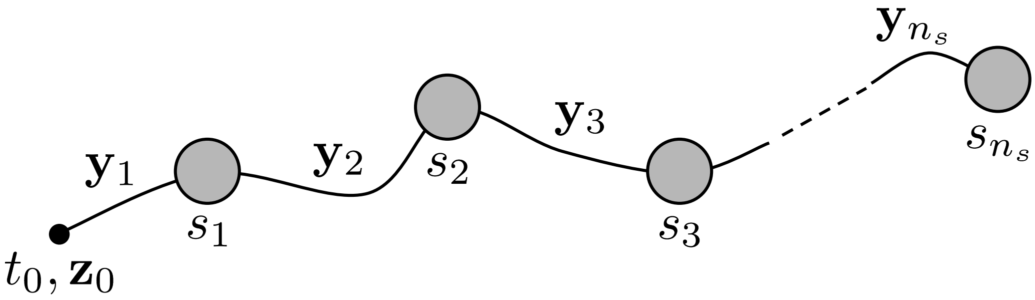


Figure 2.2: A tour trajectory and its associated decision variables. $\mathbf{z}_0$ represents the initial state at time t_0. $s_1 \ldots s_{n_s} \in O$ denote the sequence of objects visited in the tour, and $\mathbf{y}_1 \ldots \mathbf{y}_{n_s}$ give the continuous decision variables of each segment. The tour consists of n_s trajectory segments visiting n_s target objects.

Name	Description
t_0	Initial time of the tour
$\mathbf{z}_0$	Initial state of the tour at time t_0
$s_1 \ldots s_{n_s}$	Sequence of objects (from the set O) visited in the tour
$\mathbf{y}_1 \ldots \mathbf{y}_{n_s}$	Continuous decision variables of each trajectory segment (contains the duration of each segment Δt_i)

Table 2.1: Decision variables for the general tour trajectory model.

the next. A tour begins at the initial conditions $\mathbf{z}_0$ at time t_0 and continues for n_s trajectory segments visiting n_s target objects. Figure 2.2 shows a tour trajectory, and Table 2.1 summarizes the decision variables. We can combine all of the decision variables into a solution vector $\mathbf{x}$ as

$$\mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] \tag{2.10}$$

Given a solution $\mathbf{x}$, we can compute the state $\mathbf{z}$ at any time according to the problem-provided prescribed dynamics as

$$\mathbf{z}(t) = \mathbf{f}(t, \mathbf{x}) \tag{2.11}$$

A tour trajectory optimization problem can then be written in terms of a scalar objective function $J(\mathbf{x})$, dynamics $\mathbf{f}$ and inequality constraints $\mathbf{C}$ as.

$$\begin{aligned}
\text{Determine} \quad & \mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] && (2.12)\\
\text{minimizing} \quad & J(\mathbf{x}) && (2.13)\\
\text{subject to} \quad & \mathbf{z}(t) = \mathbf{f}(t, \mathbf{x}) && (2.14)\\
& \mathbf{C}_{initial}(t_0, \mathbf{z}_0) \leq \mathbf{0} && (2.15)\\
& \mathbf{C}_{sequence}(s_1 \ldots s_{n_s}) \leq \mathbf{0} && (2.16)\\
& \mathbf{C}_{segment}(s_i, \mathbf{y}_i) \leq \mathbf{0} \qquad \text{for } i = 1 \ldots n_s && (2.17)\\
& \mathbf{C}(\mathbf{x}) \leq \mathbf{0} && (2.18)
\end{aligned}$$

Note that equality constraints can be expressed as two inequality constraints in the above formulation. The definitions of the objectives, dynamics and constraints above are problem specific. Separable constraints of the initial conditions, tour sequence, or segment decision variables may exist (Equations (2.15) through (2.17)). Constraints coupling all of the decision may also exist (Equation (2.18)). This model is general in the sense that it encompasses problems of simpler types. For example, when the dynamics vanish and the objective is to minimize the total travel cost visiting all target objects, it represents a traveling salesman problem. Alternatively, when the object sequence is fixed, we have a continuous trajectory optimization problem. The objective, dynamics and constraints in the model may be nonlinear, and the sequence decision variables $s_1 \ldots s_{n_s}$ can represented as a set integer decision variables. The model is therefore a mixed-integer nonlinear programming problem [54, 22].

m_0	Initial mass (with fuel)
m_{dry}	Dry mass (fuel exhausted)
T_{max}	Maximum thrust magnitude
I_{sp}	Specific impulse

Table 2.2: Spacecraft parameters and descriptions.

2.2 Spacecraft Tour Trajectories

We now implement the general tour model described in the previous section for the specific case of spacecraft tour trajectories. The agent of the model is now a spacecraft, and the target objects are now objects in space such as satellites, asteroids or other celestial objects. We treat the spacecraft and target objects as point masses throughout the development.

2.2.1 Spacecraft Dynamics

We define the spacecraft with a propulsion system and a corresponding mass of fuel onboard with which to make maneuvers. We limit our development to a constant specific impulse propulsion system with a maximum thrust magnitude [16]. Table 2.2 gives the parameters of the spacecraft. We extend to the state vector $\mathbf{z}$ to now include the spacecraft's mass as

$$\mathbf{z}(t) = \begin{pmatrix} \mathbf{r}(t) \\ \mathbf{v}(t) \\ m(t) \end{pmatrix} \tag{2.19}$$

Then, at the initial time of the tour the spacecraft's mass is

$$m(t_0) = m_0 \tag{2.20}$$

and it is constrained for all time by the limited fuel mass as

$$m(t) \geq m_{dry} \tag{2.21}$$

The spacecraft may thrust in any direction and with any magnitude up to the limit of T_{max} as long as fuel is available. We define the spacecraft's thrusting over time with $\mathbf{T}(t)$ such that $T(t) \leq T_{max}$. The spacecraft is subject to a gravitational acceleration $\mathbf{g}(\mathbf{r})$, and its motion is governed by

$$\mathbf{z}_0(t_0) = \begin{pmatrix} \mathbf{r}(t_0) \\ \mathbf{v}(t_0) \\ m(t_0) \end{pmatrix} \qquad \dot{\mathbf{z}}(t) = \begin{pmatrix} \mathbf{v}(t) \\ \mathbf{g}(\mathbf{r}(t)) + \frac{\mathbf{T}(t)}{m(t)} \\ -\frac{T(t)}{I_{sp}\, g_0} \end{pmatrix} \tag{2.22}$$

where g_0 is the standard gravitational acceleration on Earth's surface. This initial value problem can be numerically integrated given a definition of $\mathbf{g}(\mathbf{r})$ and a thrust history $\mathbf{T}(t)$. If we have Keplerian two-body motion about a central body with gravitational parameter μ, then the spacecraft's equations of motion are

$$\mathbf{z}_0(t_0) = \begin{pmatrix} \mathbf{r}(t_0) \\ \mathbf{v}(t_0) \\ m(t_0) \end{pmatrix} \qquad \dot{\mathbf{z}}(t) = \begin{pmatrix} \mathbf{v}(t) \\ -\frac{\mu}{r(t)^3}\, \mathbf{r}(t) + \frac{\mathbf{T}(t)}{m(t)} \\ -\frac{T(t)}{I_{sp}\, g_0} \end{pmatrix} \tag{2.23}$$

This ignores any effect of target objects on the spacecraft's motion. We will assume Keplerian two-body motion for the rest of the development. The spacecraft is in ballistic motion when $\mathbf{T}(t) = \mathbf{0}$; in that case it is only subject to the gravitational acceleration of the central body. The ballistic motion can then be found as the solution to Kepler's problem [8].

2.2.2 Target Object Dynamics

The motion of the target objects O_i is assumed to be known such that their state $\mathbf{X}_{O_i}$ at any time t can be represented as

$$\mathbf{X}_{O_i}(t) = \begin{pmatrix} \mathbf{r}_{O_i}(t) \\ \mathbf{v}_{O_i}(t) \end{pmatrix} \tag{2.24}$$

The state $\mathbf{X}_{O_i}(t)$ may be provided in the form of a pre-computed ephemeris and retrieved directly, or may be computed according to prescribed dynamics. For example, if the objects O_i move according to a gravitational acceleration $\mathbf{g}(\mathbf{r})$, then their states may be found at any time as the solution of the initial value problem

$$\mathbf{X}_{O_i}(t_0) = \begin{pmatrix} \mathbf{r}_{O_i}(t_0) \\ \mathbf{v}_{O_i}(t_0) \end{pmatrix} \quad \dot{\mathbf{X}}_{O_i}(t) = \begin{pmatrix} \mathbf{v}_{O_i}(t) \\ \mathbf{g}(\mathbf{r}_{O_i}(t)) \end{pmatrix} \tag{2.25}$$

which can be numerically integrated. If we further assume that the target objects O_i are in two-body Keplerian motion about a central body as we did with the spacecraft, then we have

$$\mathbf{X}_{O_i}(t_0) = \begin{pmatrix} \mathbf{r}_{O_i}(t_0) \\ \mathbf{v}_{O_i}(t_0) \end{pmatrix} \quad \dot{\mathbf{X}}_{O_i}(t) = \begin{pmatrix} \mathbf{v}_{O_i}(t) \\ -\frac{\mu}{r_{O_i}(t)^3}\,\mathbf{r}_{O_i}(t) \end{pmatrix} \tag{2.26}$$

This motion may also be integrated numerically, but can also more simply be solved as the solution to Kepler's problem [8].

2.2.3 Finite Burn Trajectory Segments

We can model the thrust history of the spacecraft $\mathbf{T}(t)$ directly as a series of finite burn maneuvers. A single finite burn maneuver models a continuous thrust over a non-zero period of time, and is therefore a realistic model for low-thrust spacecraft.

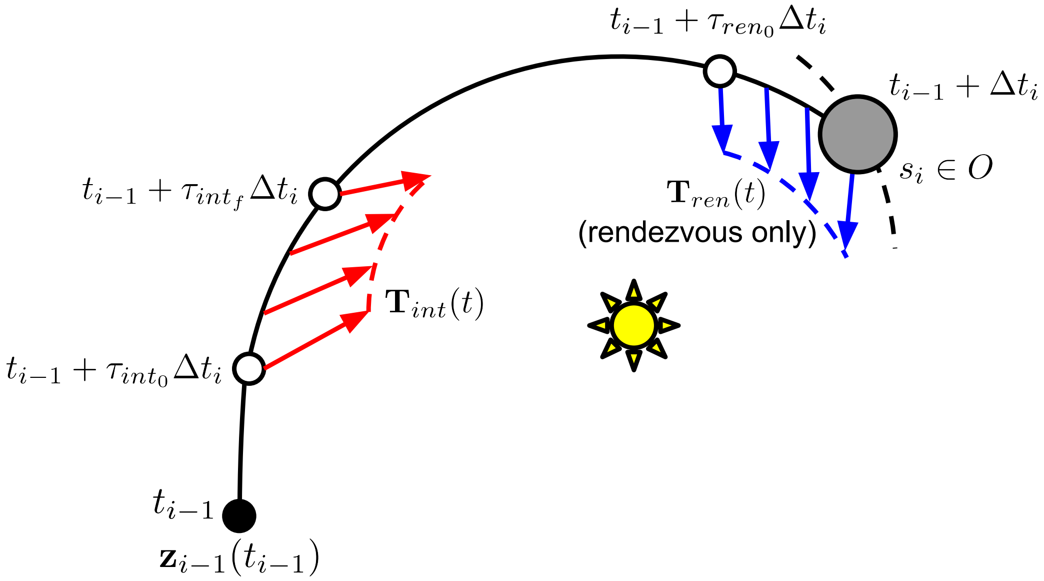

$t_{i-1} + \tau_{ren_0}\Delta t_i$
$t_{i-1} + \Delta t_i$
$s_i \in O$
$t_{i-1} + \tau_{int_f}\Delta t_i$
$\mathbf{T}_{ren}(t)$
(rendezvous only)
$\mathbf{T}_{int}(t)$
$t_{i-1} + \tau_{int_0}\Delta t_i$
t_{i-1}
$\mathbf{z}_{i-1}(t_{i-1})$

Figure 2.3: A finite burn trajectory segment. An initial maneuver causes the spacecraft to intercept the target object, and an optional final maneuver causes a rendezvous.

Here we will define trajectory segments using finite burns for use in constructing tour trajectories.

Figure 2.3 gives one possible parameterization for a finite burn segment. There are two finite burn maneuvers: $\mathbf{T}_{int}(t)$ intercepting the target object and $\mathbf{T}_{ren}(t)$ completing an optional rendezvous to match its velocity. The τ decision variables give time bounds on the finite burn maneuvers and are constrained such that

$$0 \leq \tau_{int_0} < \tau_{int_f} \leq \tau_{ren_0} < 1 \tag{2.27}$$

The rendezvous maneuver ends at the final time of the segment. In order to parameterize the thrust histories, let us decompose the thrust vector $\mathbf{T}(t)$ into its magnitude $T(t)$ and a unit thrust direction $\hat{\mathbf{u}}(t)$, such that

$$\mathbf{T}(t) = T(t)\ \hat{\mathbf{u}}(t) \tag{2.28}$$

$$0 \leq T(t) \leq T_{max} \tag{2.29}$$

$$\|\hat{\mathbf{u}}(t)\| = 1 \tag{2.30}$$

The unit thrust direction can be further parameterized by spherical angles $\alpha(t)$ and $\beta(t)$ as

$$\hat{\mathbf{u}}(t) = \begin{pmatrix} \cos\alpha(t)\cos\beta(t) \\ \sin\alpha(t)\cos\beta(t) \\ \sin\beta(t) \end{pmatrix} \tag{2.31}$$

If we assume a quadratic steering model [49], then we have

$$\alpha(t) = \alpha_0 + \dot{\alpha}_0(t - t_{fb_0}) + \frac{\ddot{\alpha}_0(t - t_{fb_0})^2}{2} \tag{2.32}$$

$$\beta(t) = \beta + \dot{\beta}_0(t - t_{fb_0}) + \frac{\ddot{\beta}_0(t - t_{fb_0})^2}{2} \tag{2.33}$$

where t_{fb_0} is the beginning time of the maneuver. Finally, we can fix the thrust magnitude to its maximum value such that

$$T(t) = T_{max} \tag{2.34}$$

The continuous decision variables for a finite burn trajectory segment intercepting a target object are then

$$\mathbf{y}_i^{int} = \begin{pmatrix} \Delta t_i & \tau_{int_0} & \tau_{int_f} & (\alpha_0 & \dot{\alpha}_0 & \ddot{\alpha}_0 & \beta_0 & \dot{\beta}_0 & \ddot{\beta}_0)_{int} \end{pmatrix}^T_{1\times 9} \tag{2.35}$$

and for the rendezvous case they are

$$\mathbf{y}_i^{ren} = \begin{pmatrix} \Delta t_i & \tau_{int_0} & \tau_{int_f} & (\alpha_0 & \dot{\alpha}_0 & \ddot{\alpha}_0 & \beta_0 & \dot{\beta}_0 & \ddot{\beta}_0)_{int} & \cdots \\ \tau_{ren_0} & (\alpha_0 & \dot{\alpha}_0 & \ddot{\alpha}_0 & \beta_0 & \dot{\beta}_0 & \ddot{\beta}_0)_{ren} & & & \end{pmatrix}^T_{1\times 16} \tag{2.36}$$

The segment decision variables $\mathbf{y}_i$ must then be chosen to intercept or rendezvous with the target object s_i, given the initial conditions $\mathbf{z}_{i-1}$ at t_{i-1}. For each segment in a tour then, the segment conditions that must be satisfied for the intercept case are

$$\mathbf{C}(\mathbf{y}_i^{int}) = \Big(\mathbf{r}(t_{i-1} + \Delta t_i) - \mathbf{r}_{s_i}(t_{i-1} + \Delta t_i)\Big)_{3\times 1} = \mathbf{0} \tag{2.37}$$

and for the rendezvous case are

$$\mathbf{C}(\mathbf{y}_i^{ren}) = \begin{pmatrix} \mathbf{r}(t_{i-1} + \Delta t_i) - \mathbf{r}_{s_i}(t_{i-1} + \Delta t_i) \\ \mathbf{v}(t_{i-1} + \Delta t_i) - \mathbf{v}_{s_i}(t_{i-1} + \Delta t_i) \end{pmatrix}_{6\times 1} = \mathbf{0} \tag{2.38}$$

In both cases we have an underdetermined system of nonlinear equations. Multiple feasible solutions may be possible, or a feasible solution may not exist. If multiple feasible solutions are possible, then a single solution can be found by defining an objective function and computing an optimal solution. However, multiple locally optimal solutions may also exist. In general this is a challenging the problem, and constructing a tour trajectory composed of multiple of these segments only increases the difficulty. Existing systems such as Copernicus can be used to find such solutions given a fixed sequence $s_1 \ldots s_{n_s}$ [51, 71]. Olympio provides an optimal control formulation of the tour problem for fixed sequences as well [52].

2.2.4 Impulsive Trajectory Segments

We can alternatively develop trajectory segments in terms of impulsive maneuvers, ignoring the spacecraft's limited maximum thrust magnitude. An impulsive maneuver occurs instantaneously and imparts a change in velocity denoted as $\Delta\mathbf{V}$. This is a discontinuous change in the spacecraft's state, and is thus a less realistic model for spacecraft, especially those with low-thrust propulsion systems where the thrust durations are long. However, this approach has advantages compared to the finite burn approach especially in the context of tour trajectories.

The states immediately before and after the maneuver are $\mathbf{z}(t^-)$ and $\mathbf{z}(t^+)$ and can be expressed as

$$\mathbf{z}(t^-) = \begin{pmatrix} \mathbf{r}(t^-) \\ \mathbf{v}(t^-) \\ m(t^-) \end{pmatrix} \quad \mathbf{z}(t^+) = \begin{pmatrix} \mathbf{r}(t^+) \\ \mathbf{v}(t^+) \\ m(t^+) \end{pmatrix} \tag{2.39}$$

The change in velocity $\Delta\mathbf{V}$ defines the maneuver. There is also a corresponding change in mass Δm corresponding to fuel used during the maneuver. The states instantaneously before and after the maneuver satisfy the relation

$$\mathbf{z}(t^+) = \mathbf{z}(t^-) + \begin{pmatrix} \mathbf{0} \\ \Delta\mathbf{V} \\ \Delta m \end{pmatrix} \tag{2.40}$$

Only the position of the spacecraft is the same across the impulse. Given the magnitude of a maneuver ΔV, the change in mass Δm may be determined using the Tsiolkovsky rocket equation that relates the magnitude of the maneuver to the propellant mass consumed [59].

$$\Delta V = I_{sp}\, g_0\, ln \left(\frac{m(t^-)}{m(t^+)} \right) \tag{2.41}$$

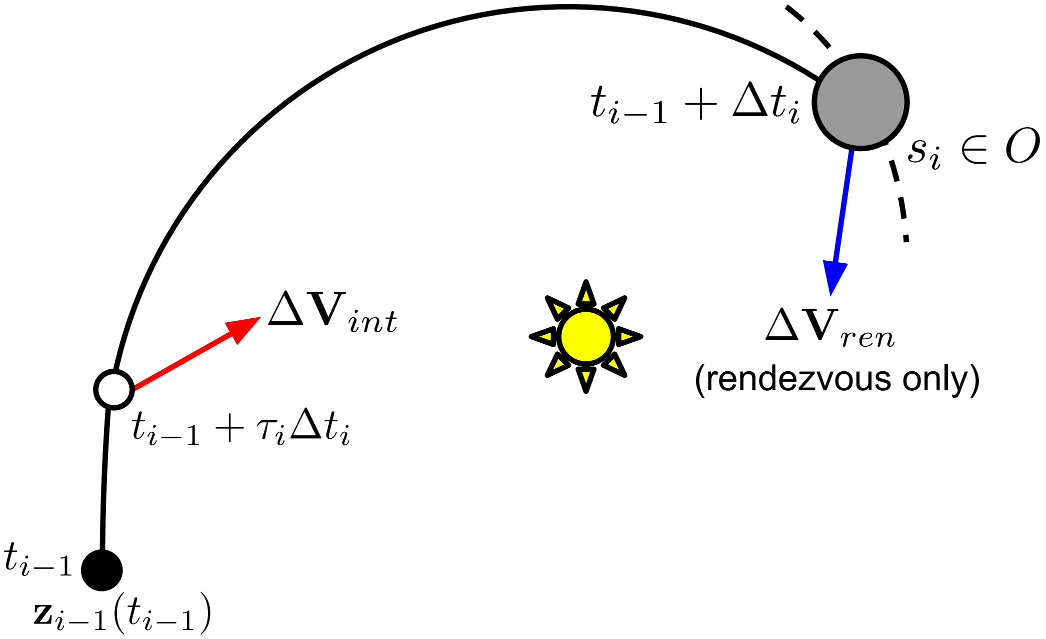

$t_{i-1} + \Delta t_i$
$s_i \in O$
$\Delta \mathbf{V}_{int}$
$\Delta \mathbf{V}_{ren}$
(rendezvous only)
$t_{i-1} + \tau_i \Delta t_i$
t_{i-1}
$\mathbf{z}_{i-1}(t_{i-1})$

Figure 2.4: An impulsive trajectory segment. An initial impulse $\Delta \mathbf{V}_{int}$ maneuvers the spacecraft to intercept the target object, and an optional final impulse $\Delta \mathbf{V}_{ren}$ maneuvers the spacecraft to rendezvous with the target.

Manipulating the rocket equation we then have

$$\Delta m = m(t^+) - m(t^-) = m(t^-)\left(e^{-\Delta V/(I_{sp}g_0)} - 1\right) \tag{2.42}$$

The state change that occurs for an impulsive maneuver is therefore completely determined by the $\Delta \mathbf{V}$ maneuver as

$$\mathbf{z}(t^+) = \mathbf{z}(t^-) + \begin{pmatrix} \mathbf{0} \\ \Delta \mathbf{V} \\ m(t^-)\left(e^{-\Delta V/(I_{sp}g_0)} - 1\right) \end{pmatrix} \tag{2.43}$$

Figure 2.4 shows an impulsive trajectory segment. The segment begins at a given initial state $\mathbf{z}_{i-1}$ at t_{i-1} and has a total duration Δt_i. An intercept maneuver

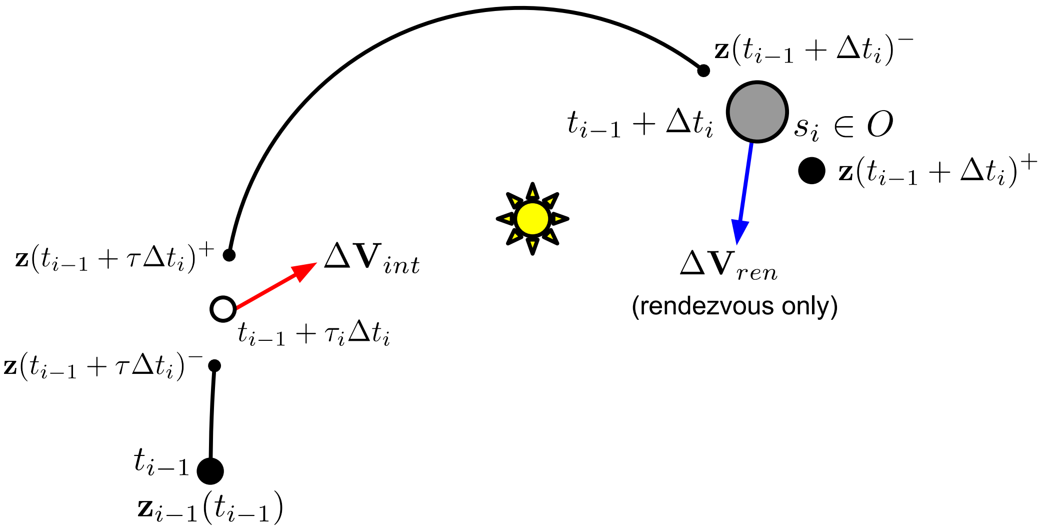

$\mathbf{z}(t_{i-1}+\Delta t_i)^-$
$t_{i-1}+\Delta t_i$
$s_i \in O$
$\mathbf{z}(t_{i-1}+\Delta t_i)^+$
$\mathbf{z}(t_{i-1}+\tau\Delta t_i)^+$
$\Delta\mathbf{V}_{int}$
$\Delta\mathbf{V}_{ren}$
(rendezvous only)
$t_{i-1}+\tau_i\Delta t_i$
$\mathbf{z}(t_{i-1}+\tau\Delta t_i)^-$
t_{i-1}
$\mathbf{z}_{i-1}(t_{i-1})$

Figure 2.5: The impulsive trajectory segment shown in Figure 2.4 with all spacecraft states and discontinuities shown.

occurs at time $t_{i-1} + \tau_i \Delta t_i$, where $0 \leq \tau_i \leq 1$. Finally, an optional maneuver to rendezvous with the target object occurs at $t_{i-1} + \Delta t_i$. The states of the spacecraft are discontinuous across the maneuvers, and are shown schematically in Figure 2.5.

Recall that the initial state of the spacecraft $\mathbf{z}_{i-1}(t_{i-1})$ is known, and the states of the target objects are known for all time. The targeting problem is therefore to determine the $\Delta\mathbf{V}$ maneuvers such that

$$\mathbf{r}(t_{i-1}+\Delta t_i) - \mathbf{r}_{O_i}(t_{i-1}+\Delta t_i) = \mathbf{0} \quad \text{(intercept)} \tag{2.44}$$

$$\begin{pmatrix} \mathbf{r}(t_{i-1}+\Delta t_i) - \mathbf{r}_{O_i}(t_{i-1}+\Delta t_i) \\ \mathbf{v}(t_{i-1}+\Delta t_i) - \mathbf{v}_{O_i}(t_{i-1}+\Delta t_i) \end{pmatrix} = \mathbf{0} \quad \text{(rendezvous)} \tag{2.45}$$

For two-body Keplerian motion, these maneuvers may be computed as a solution to Lambert's problem [9, 64]. Concisely, given two positions and the elapsed time between them, the solutions of Lambert's problem provide the velocities at the endpoints.

$$\textsc{Lambert}(\mathbf{r}_1, \mathbf{r}_2, \Delta t) \implies \mathbf{v}_1, \mathbf{v}_2 \tag{2.46}$$

These computed velocities allow for the determination of $\Delta\mathbf{V}_{int}$ and $\Delta\mathbf{V}_{ren}$ in the impulsive trajectory segment. It is important to note that in general there are multiple solutions to Lambert's problem corresponding to "short way", "long way", and multiple revolution solutions [28]. It is assumed that the solution corresponding to the minimum total maneuver magnitude is used. Thus, for the intercept we only need to know when the maneuver occurs; a rendezvous maneuver occurs at the time of intercept to match velocity with the target.

We can now define the parameters for the impulsive trajectory segment as the segment duration and intercept maneuver time

$$\mathbf{y}_i = \begin{pmatrix} \Delta t_i \\ \tau_i \end{pmatrix} \tag{2.47}$$

These uniquely determine the maneuvers and trajectory to the target object s_i. The segment can be fully computed given its initial conditions, target object and segment decision variables. The resulting computation gives the values for any intercept and rendezvous maneuvers as

$$\mathbf{z}_i(t_i) = \mathbf{z}_i(t_{i-1}, \mathbf{z}_{i-1}, s_i, \mathbf{y}_i) \implies \Delta\mathbf{V}_{int_i}, \Delta\mathbf{V}_{ren_i} \tag{2.48}$$

2.2.5 Impulsive Maneuver to Finite Burn Maneuver Conversion

The use of impulsive maneuvers simplifies the generation of tour trajectories. We are able to parameterize an impulsive trajectory segment with only one additional decision variable: the time at which the intercept maneuver occurs. We can then compute the resulting intercept and optional rendezvous maneuvers by solving the corresponding Lambert's problem. In contrast, a finite burn trajectory segment is

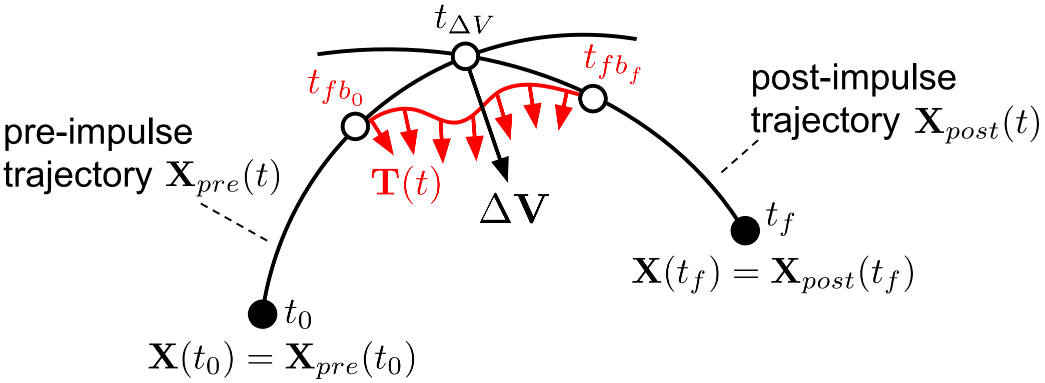

$t_{\Delta V}$
t_{fb_0}
t_{fb_f}
pre-impulse trajectory $\mathbf{X}_{pre}(t)$
post-impulse trajectory $\mathbf{X}_{post}(t)$
$\mathbf{T}(t)$
$\Delta\mathbf{V}$
t_f
$\mathbf{X}(t_f) = \mathbf{X}_{post}(t_f)$
t_0
$\mathbf{X}(t_0) = \mathbf{X}_{pre}(t_0)$

Figure 2.6: A valid finite burn representation of an impulsive $\Delta\mathbf{V}$ maneuver The finite burn begins at time t_{fb_0}, thrusts over a period of time and matches position and velocity with the post-impulse trajectory at time t_{fb_f} [50].

more difficult; parameterizations require many more decision variables that must be chosen to satisfy intercept or rendezvous conditions. This can be an iterative procedure, and there is no guarantee that a feasible solution will be found. Finite burn trajectory segments, however, realistically model low-thrust trajectories and are therefore more useful for practical applications. This section describes the conversion of impulsive maneuvers to equivalent finite burns and develops constraints on impulsive maneuvers such that the conversion is likely to be feasible. This in turn allows us to use impulsive trajectory segments in the search for optimal tour trajectories and then later convert an impulsive solution to an equivalent finite burn solution if necessary.

Figure 2.6 shows a finite burn maneuver representation of an impulsive maneuver [50]. The impulsive $\Delta\mathbf{V}$ occurs at $t_{\Delta V}$ and instantaneously changes the spacecraft's trajectory from the pre-impulse trajectory $\mathbf{X}_{pre}(t_{\Delta V})$ to the post-impulse trajectory $\mathbf{X}_{post}(t_{\Delta V})$. A valid finite burn representation instead maneuvers from the pre-impulse spacecraft trajectory to the post-impulse trajectory over a non-zero period of time. The finite burn begins at time t_{fb_0} and continues until it matches position and velocity with the post-impulse trajectory at time t_{fb_f}. Determining such a finite burn maneuver requires finding values for t_{fb_0}, t_{fb_f} and the thrust time history $\mathbf{T}(t)$ such that $\mathbf{X}(t_{fb_f}) = \mathbf{X}_{post}(t_{fb_f})$. Rather than treating the full conversion process, we instead attempt only to find constraints on the impulsive maneuvers such that the conversion may be feasible. For the above finite burn to be valid, its burn time must not fall outside of the prescribed bounds t_0 and t_f.

$$t_0 \leq t_{fb_0} < t_{fb_f} \leq t_f \tag{2.49}$$

So far the time bounds t_0 and t_f are arbitrary. We could define them to correspond to the bounds of a trajectory segment in a tour, or define them in relation to other proximal finite burn maneuvers.

We can estimate the duration a finite burn representation of an impulsive $\Delta\mathbf{V}$ based on the rocket equation. If we assume the spacecraft thrusts at T_{max} for the duration of the finite burn, then the predicted burn time is

$$\Delta t_{\Delta V} = -\frac{\Delta m}{T_{max}/\left(I_{sp}\ g_0\right)} = \frac{m(t^-)\left(1 - e^{-\Delta V/(I_{sp} g_0)}\right)}{T_{max}/\left(I_{sp}\ g_0\right)} \tag{2.50}$$

For both impulsive and finite burn trajectory segments, we have described two types of maneuvers: intercepts and rendezvouses. For the intercept case, we assume the impulse to be at the midpoint of the corresponding finite burn. For the rendezvous case, we assume that the finite burn ends at the time of the impulse. The impulsive maneuvers and their corresponding finite burn representations are shown in Figure 2.7. The estimated start times for finite burn representations of intercept and

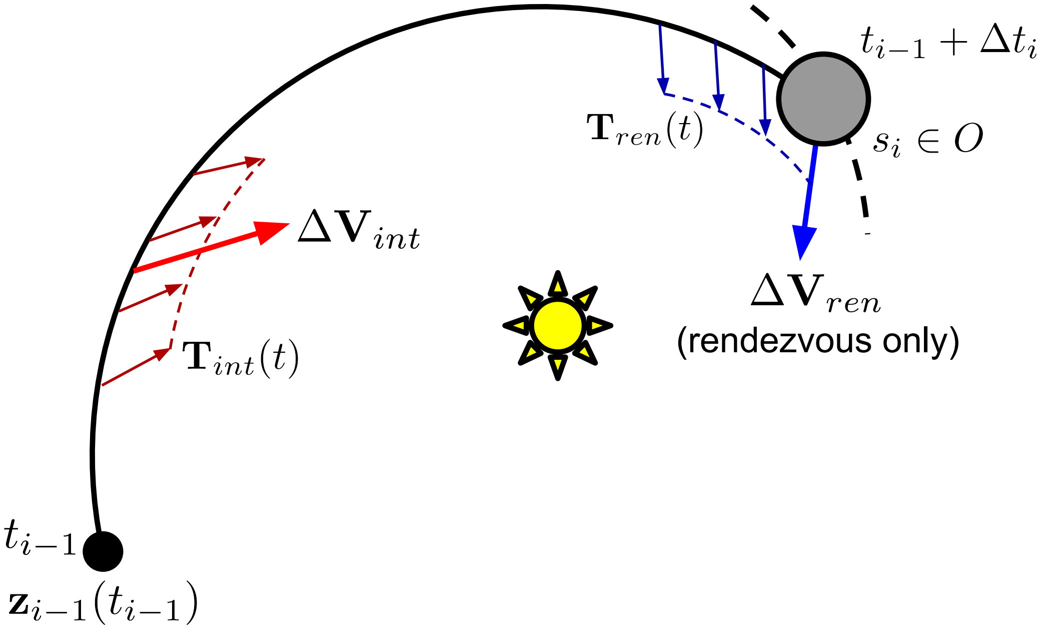


Figure 2.7: An impulsive trajectory segment with estimates for finite burn maneuvers replacing the impulses.

rendezvous maneuvers are then

$$t_{fb_0} = t_{\Delta V} - \frac{\Delta t_{\Delta V}}{2} \qquad \text{(intercept)} \qquad (2.51)$$

$$t_{fb_0} = t_{\Delta V} - \Delta t_{\Delta V} \qquad \text{(rendezvous)} \qquad (2.52)$$

In both cases, the end time is

$$t_{fb_f} = t_{\Delta V} + \Delta t_{\Delta V} \qquad (2.53)$$

We can add constraints on impulsive tours that prevent the finite burn estimates of their impulsive maneuvers from overlapping in time. This additionally restricts the feasible solution space so that the solutions found are more likely to have corresponding feasible finite burn trajectories. The constraints are in terms of the time bounds of Equations 2.51 through 2.53. If we have a set of $N_{\Delta V}$ maneuvers in a tour trajectory (intercepts and/or rendezvouses), then the additional constraints are

$$(t_{fb_0})_0 \geq t_0 \qquad (2.54)$$

$$(t_{fb_f})_{N_{\Delta V}} \leq t_{n_s} \qquad (2.55)$$

$$(t_{fb_f})_i \leq (t_{fb_0})_{i+1} \qquad \text{for } i \in 1 \ldots N_{\Delta V} - 1 \qquad (2.56)$$

Equations (2.54) and (2.55) constrain the finite burn maneuvers from occuring outside of the initial and final times of the full tour trajectory, respectively. Equation (2.56) constrains the finite burn maneuvers from overlapping each other. Constructing an initial guess for the thrust histories $\mathbf{T}(t)$ is beyond the scope of the current work, but is addressed by Ocampo [50].

Gravity Losses

The estimate for the duration of a finite burn given in Equation 2.50 is based on the Tsiolkovsky rocket equation, and is thus only exact in the absence of external forces. We now consider performance penalties associated with the replacement of impulsive maneuvers with finite burns in the presence of gravity, following the analysis of Robbins [57]. Adjusting our finite burn estimates for these penalties will tighten the associated constraints on the impulsive maneuvers and improve the feasibility of the impulsive to finite burn conversion.

We define the characteristic velocity of a finite burn maneuver as the integral of the thrust acceleration magnitude over the duration of the maneuver. Assuming continuous maximum thrust, we have

$$\Delta V_{fb} = \int_{t_{fb_0}}^{t_{fb_f}} \frac{T_{max}}{m(t)} dt \tag{2.57}$$

When gravity (or more generally a gravity gradient) is absent and the thrust direction is constant, the characteristic velocity ΔV_{fb} is equal to the total velocity change achieved, and thus the finite burn estimate is an exact replacement for the corresponding impulsive maneuver. However, in realistic cases where a gravity gradient exists there is a performance penalty associated with the use of finite thrust such that

$$\Delta V_{fb} > \Delta V \tag{2.58}$$

That is, the characteristic velocity of the finite burn maneuver must be higher than the corresponding impulsive ΔV to achieve the same result. The finite burn maneuver must then thrust over a longer period of time using a larger mass of fuel. Robbins provides an analytical upper bound for this penalty as

$$\Delta V_{fb} \leq \Delta V + \frac{1}{24}(\omega_s \Delta t_{\Delta V})^2 \Delta V \tag{2.59}$$

where $\Delta t_{\Delta V}$ is our original rocket equation estimate for the finite burn duration, and ω_s is the Schuler frequency defined as

$$\omega_s^2 = \frac{\mu}{r^3} \tag{2.60}$$

This approximate upper bound is valid for mass-optimal impulsive maneuvers and cases where the dimensionless quantity $\omega_s \Delta t_{\Delta V}$ does not exceed unity. For a spacecraft orbiting the sun at 1 AU, the approximation is therefore valid for finite burn durations up to approximately 60 days. We see that the penalty grows rapidly with the duration of the finite burn maneuver as expected.

We now use this upper bound in our impulsive trajectory segment computations. We use

$$\Delta V_{fb} = \Delta V + \frac{1}{24}(\omega_s \Delta t_{\Delta V})^2 \Delta V \tag{2.61}$$

We replace the change in mass across the impulsive maneuver with the estimated mass use of the representative finite burn,

$$\Delta m_{fb} = m(t^-)\left(e^{-\Delta V_{fb}/(I_{sp} g_0)} - 1\right) \tag{2.62}$$

The new change in mass corresponds to a longer finite burn time which we compute as

$$\Delta t_{fb} = -\frac{\Delta m_{fb}}{T_{max}/\left(I_{sp}\, g_0\right)} \tag{2.63}$$

We now modify the state change across an impulsive maneuver to correspond to the finite burn estimate's mass usage as

$$\mathbf{z}(t^+) = \mathbf{z}(t^-) + \begin{pmatrix} \mathbf{0} \\ \Delta \mathbf{V} \\ \Delta m_{fb} \end{pmatrix} \tag{2.64}$$

and update the estimated bounds on intercept and rendezvous finite burn maneuvers as

$$t_{fb_0} = t_{\Delta V} - \frac{\Delta t_{fb}}{2} \qquad \text{(intercept)} \tag{2.65}$$

$$t_{fb_0} = t_{\Delta V} - \Delta t_{fb} \qquad \text{(rendezvous)} \tag{2.66}$$

$$t_{fb_f} = t_{\Delta V} + \Delta t_{fb} \tag{2.67}$$

Compensating for gravity losses yields higher characteristic velocities for the finite burn maneuvers and a corresponding increase in fuel mass usage. It is interesting to note however that the effects are greater on earlier maneuvers in a trajectory. The increased fuel mass usage (and therefore reduced mass) makes the spacecraft more efficient for later maneuvers. In fact, the timing constraints on the finite burn maneuvers can actually be less restrictive later in a tour than the ideal case ignoring gravity losses. Figure 2.8 shows the ratios of the characteristic velocities and finite burn durations with gravity losses versus the ideal case for 50 sequential maneuvers. While the finite burn characteristic velocity always exceeds the impulsive value, we see that after a number of maneuvers the estimated maneuver duration becomes less than when gravity losses are not considered. The associated timing constraints would then be less restrictive than the ideal case where gravity losses are ignored.

Finally, we note that these finite burn estimates and their associated constraints are only valid for mass-optimal impulses and short finite burn durations. The additional constraints therefore do not guarantee feasible finite burn conversions. However, they do restrict the feasible solution space to improve the likelihood that such a conversion is possible.

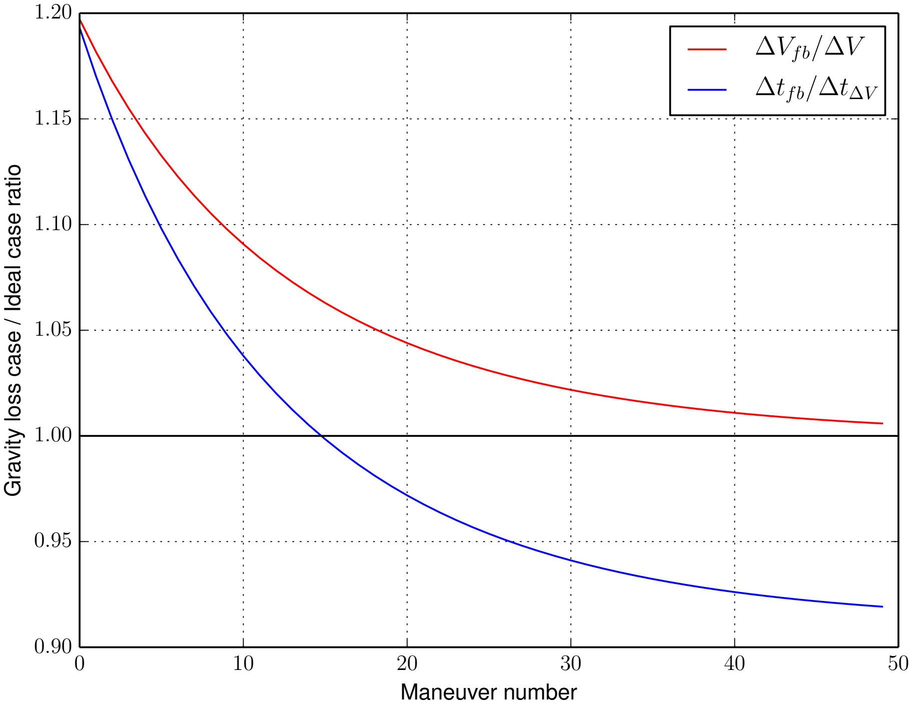


Figure 2.8: The finite burn characteristic velocity and duration ratios are shown for the case where gravity losses are considered and for the ideal case. 50 consecutive maneuvers are made for an impulsive ΔV of 1 km/s. The spacecraft parameters correspond to the GTOC4 problem described in Chapter 5.

2.2.6 Augmented Impulsive Tour Model

We implement the general tour trajectory model of Section 2.1.4 for spacecraft tour trajectories. The state of the spacecraft includes its mass such that

$$\mathbf{z}(t) = \begin{pmatrix} \mathbf{r}(t) \\ \mathbf{v}(t) \\ m(t) \end{pmatrix} \tag{2.68}$$

We use the impulsive model for trajectory segments and maneuvers. The continuous segment decision variables are

$$\mathbf{y}_i = \begin{pmatrix} \Delta t_i \\ \tau_i \end{pmatrix} \tag{2.69}$$

The required impulsive maneuvers are computed by solving Lambert's problem, and the ballistic motion is found by solving Kepler's problem. The final state of each segment and the impulsive maneuvers can be found as

$$\mathbf{z}_i(t_i) = \mathbf{z}_i(t_{i-1}, \mathbf{z}_{i-1}, s_i, \mathbf{y}_i) \implies \Delta\mathbf{V}_{int_i}, \Delta\mathbf{V}_{ren_i} \tag{2.70}$$

The final mass of the tour is restricted by the available fuel so that $m(t) \geq m_{dry}$.

We augment the model to consider the feasibility of converting the impulsive maneuvers to finite burns. The state change of the spacecraft across impulsive maneuvers is then

$$\mathbf{z}(t^+) = \mathbf{z}(t^-) + \begin{pmatrix} \mathbf{0} \\ \Delta\mathbf{V} \\ \Delta m_{fb} \end{pmatrix} \tag{2.71}$$

where Δm_{fb} is the change in mass for a representative finite burn corrected for gravity losses given in Equation 2.62. Further, we add time constraints for each impulsive maneuver such that the corresponding finite burn maneuvers do not overlap in time. These constraints are given by Equations 2.54 through 2.56 where the finite burn time bounds are given by Equations 2.65 through 2.67. The augmented impulsive tour model is thus

$$\text{Determine} \quad \mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] \tag{2.72}$$

$$\text{minimizing} \quad J(\mathbf{x}) \tag{2.73}$$

$$\text{subject to} \quad \mathbf{z}(t) = \mathbf{f}(t, \mathbf{x}) \implies \Delta\mathbf{V}_1 \ldots \Delta\mathbf{V}_{N_{\Delta V}} \tag{2.74}$$

$$m(t_{n_s}) \geq m_{dry} \tag{2.75}$$

$$(t_{fb_0})_0 \geq t_0 \tag{2.76}$$

$$(t_{fb_f})_{N_{\Delta V}} \leq t_{n_s} \tag{2.77}$$

$$(t_{fb_f})_i \leq (t_{fb_0})_{i+1} \qquad \text{for } i \in 1 \ldots N_{\Delta V} - 1 \tag{2.78}$$

$$\mathbf{C}_{initial}(t_0, \mathbf{z}_0) \leq \mathbf{0} \tag{2.79}$$

$$\mathbf{C}_{sequence}(s_1 \ldots s_{n_s}) \leq \mathbf{0} \tag{2.80}$$

$$\mathbf{C}_{segment}(s_i, \mathbf{y}_i) \leq \mathbf{0} \qquad \text{for } i = 1 \ldots n_s \tag{2.81}$$

$$\mathbf{C}(\mathbf{x}) \leq \mathbf{0} \tag{2.82}$$

We still allow general constraints on the initial conditions, sequence, segment decision variables and full decision vector that can be implemented for specific problems. The definition of an objective $J(\mathbf{x})$ is also left to a problem statement.

The impulsive tour model simplifies the search for optimal trajectories. Each segment has a single additional decision variable that uniquely determines the required maneuvers and trajectory of the segment. Finite burn segments more realistically model low-thrust spacecraft, but add to the complexity of the model since a two-point boundary value problem must be solved at each segment to satisfy intercept or rendezvous conditions. The addition of the finite burn feasibility constraints to the impulsive tour model aims to ensure that the impulsive trajectory has a corresponding finite burn trajectory. This therefore allows for a two stage approach: search and optimization of an impulsive tour trajectory, followed by conversion of impulsive trajectories to realistic finite burn tours.

Chapter 3

Global Search Methodology

This chapter develops a search methodology for finding solutions to tour trajectory design and optimization problems. The approach is based on the Tabu search metaheuristic developed by Glover [27], but draws on elements from graph theory and path finding approaches including the A* search algorithm [32]. We develop the methodology in terms of building blocks common to local search approaches: the solution representation, neighborhood and objective function. First, we have a given or incumbent solution representing a tour trajectory denoted as $\mathbf{x}$. For a given problem, the solution $\mathbf{x}$ may be represented or encoded in a variety of ways; the chosen solution representation can have a significant impact on the implementation and performance of the search. Next, we define a neighborhood of solutions nearby the incumbent. This neighborhood $\mathcal{N}(\mathbf{x})$ determines the candidate solutions that can be moved to from the incumbent $\mathbf{x}$. Finally, a scalar objective function $J(\mathbf{x})$–provided in a problem statement–gives a metric for comparing solutions. We assume without loss of generality that the goal is to minimize this objective function. A simple local search algorithm illustrating these building blocks is given in Algorithm 1. The algorithm is considered greedy since at every iteration it moves to

Algorithm 1 Local search algorithm.

```
procedure LocalSearch($\mathbf{x}_0$)
    repeat
        ▷ find the best solution in the neighborhood of the incumbent
        $\mathbf{x}' \leftarrow \operatorname*{argmin}_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_0)} J(\mathbf{x})$
        ▷ accept the solution as the new incumbent if it is an improvement
        ▷ otherwise, we have converged to a locally optimal solution
        if $J(\mathbf{x}') < J(\mathbf{x}_0)$ then
            $\mathbf{x}_0 \leftarrow \mathbf{x}'$
        else
            converged
        end if
    until converged
    ▷ return the best solution found
    return $\mathbf{x}_0$
end procedure
```

the best possible candidate solution in the neighborhood. Further, it can only find locally optimal solutions since it never moves beyond its neighborhood or accepts non-improving moves. We consider this approach, however, as a basis for developing more advanced approaches that address these weaknesses.

The following sections develop the solution representation, neighborhoods, and objective functions used in the search. These are then combined to form the search algorithm.

3.1 Solution Representation

A solution representation encodes the decision variables of a problem into a form that a search or optimization algorithm can evaluate and manipulate to move to new solutions. Since all solutions considered are in terms of this representation, its definition determines the search space. For example, consider the traveling salesman problem. For an n-city problem, one solution representation is a permutation of the integers $1 \dots n$, where each number corresponds to a city and the order determines the sequence [43]. The search space consists of all of these permutations and has a size of $n!$ solutions. However, consider that this is a symmetric traveling salesman problem where the distances from cities i to j and j to i are equal. In this case, a tour and the same tour in reverse order can be considered equivalent, and the search space can be reduced by half. Further, since the problem statement requires the tour be a complete cycle, the starting city is unimportant and can be fixed. The search space can thus be reduced further to $\frac{(n-1)!}{2}$ solutions. The original solution representation yields a search space $2n$ times larger than what is required and therefore may be less efficient for solving the problem. Conversely, other solution representations may underrepresent the feasible solution space of the problem, leaving out good or even optimal solutions. A careful definition of the solution representation is critical to the performance of the search or optimization.

Section 2.1 gives a model for a general tour trajectory. The decision variables are t_0 and $\mathbf{z}_0$ representing the initial time and state, $s_1 \dots s_{n_s}$ representing the target object sequence, and $\mathbf{y}_1 \dots \mathbf{y}_{n_s}$ representing the continuous decision variables of each segment of the sequence. Recall a solution is denoted as $\mathbf{x}$. The most basic solution representation would be a concatenation of these decision variables, for example

$$\mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \dots s_{n_s},\ \mathbf{y}_1 \dots \mathbf{y}_{n_s}] \tag{3.1}$$

This solution representation gives the information needed to represent a full tour trajectory. However, other solution representations are possible.

3.1.1 Properties of Tour Trajectories

There are properties associated with tour trajectories that should be noted and considered in determining what solution representation to use. First, we note that the evaluation of a single tour trajectory can be computationally expensive. While evaluating a solution to the traveling salesman problem is quick, only involving basic arithmetic, the evaluation of a tour trajectory solution is much more complex. The evaluation may involve numerically integrating equations of motion according to prescribed dynamics, computing maneuvers to target objects and other computationally expensive operations. The result is that in a fixed amount of time, we are able to compute far fewer tour trajectory evaluations than for other problems such as the traveling salesman. This translates to fewer iterations for the same search algorithm. The expense of tour evaluations should therefore be addressed in the design of the solution representation if possible.

Tour trajectories can naturally be decomposed sequentially by trajectory segments. A segment i can be completely computed given its initial state $\mathbf{z}_{i-1}$ (the final state of the previous segment) and its segment decision variables s_i and $\mathbf{y}_i$. The final state for segment i is computed functionally as

$$\mathbf{z}_i = \mathbf{z}_i(\mathbf{z}_{i-1}, s_i, \mathbf{y}_i) \tag{3.2}$$

This is a recursive relationship, since the final state of every segment depends upon the final state of the previous segment. For example, the final states at various segments can be evaluated as

$$\mathbf{z}_1 = \mathbf{z}_1(\mathbf{z}_0, s_1, \mathbf{y}_1) \tag{3.3}$$

$$\mathbf{z}_2 = \mathbf{z}_2(\mathbf{z}_1(\mathbf{z}_0, s_1, \mathbf{y}_1), s_2, \mathbf{y}_2) \tag{3.4}$$

$$\mathbf{z}_3 = \mathbf{z}_3(\mathbf{z}_2(\mathbf{z}_1(\mathbf{z}_0, s_1, \mathbf{y}_1), s_2, \mathbf{y}_2), s_3, \mathbf{y}_3) \tag{3.5}$$

Clearly, as the number of segments n_s grows, the final state of the tour trajectory $\mathbf{z}_{n_s}$ becomes more expensive to compute. Another consequence of this recursion, however, is that multiple tour trajectories can share past segments in common.

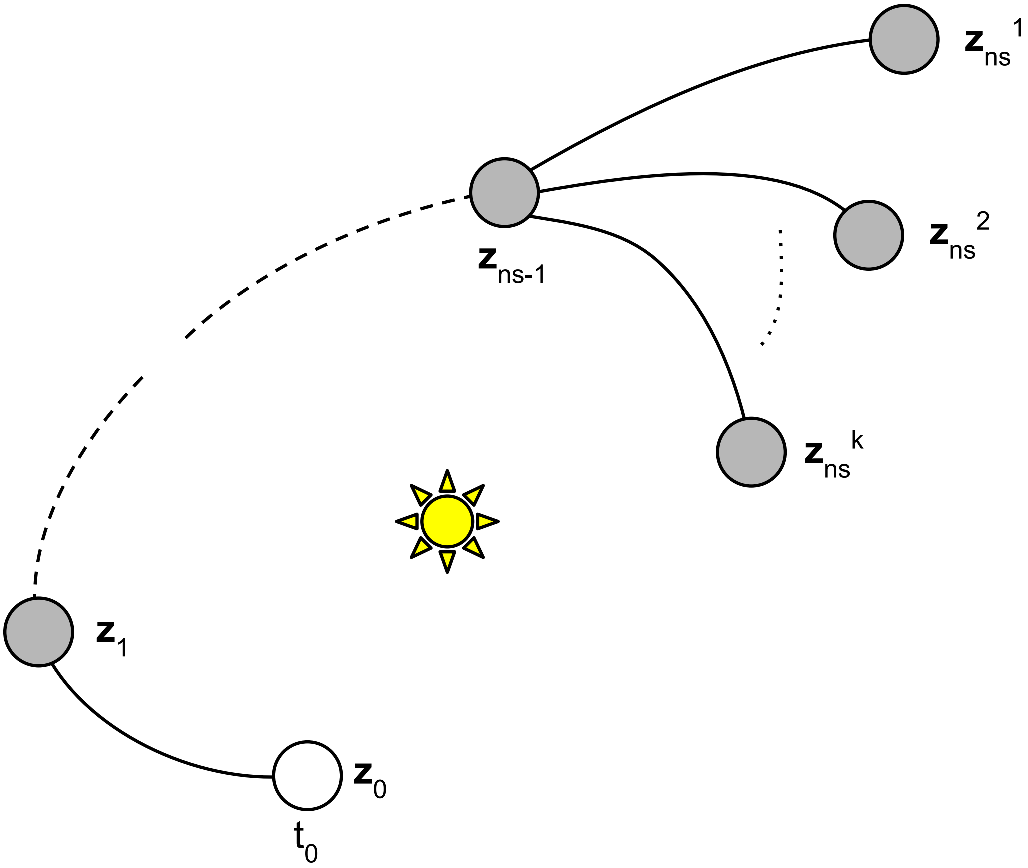


Figure 3.1: k tour trajectories are shown with final states $\mathbf{z}_{n_s}^1 \ldots \mathbf{z}_{n_s}^k$. Each trajectory shares the previous $n_s - 1$ segments in common. The state $\mathbf{z}_{n_s-1}$ can be used in the computation of the k final segments.

Consider the scenario shown in Figure 3.1. There are k tour trajectories of length n_s ending with states $\mathbf{z}_{n_s}^1 \ldots \mathbf{z}_{n_s}^k$. These k trajectories each share their first $n_s - 1$ segments in common. If the state $\mathbf{z}_{n_s-1}$ is computed once and stored, it can be reused in computing the final segments for all k trajectories, reducing the total computation time significantly (by nearly a factor of k).

Objective and constraint evaluations may also be decomposed by segment. A general objective J is a function of the entire set of decision variables $\mathbf{x}$.

$$J(\mathbf{x}) = J(t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}) \tag{3.6}$$

However, consider the objective is based in part on the total change of state of the spacecraft $\Delta\mathbf{z}$, where

$$\Delta\mathbf{z} = \mathbf{z}_{n_s} - \mathbf{z}_0 \tag{3.7}$$

This could correspond to total fuel mass used during the tour, for example, and would be commonly minimized or constrained in a tour problem. Clearly $\Delta\mathbf{z}$ can be decomposed by segment as

$$\Delta\mathbf{z} = \sum_{i=1}^{n_s} (\mathbf{z}_i - \mathbf{z}_{i-1}) = \sum_{i=1}^{n_s} \Delta\mathbf{z}_i \tag{3.8}$$

Other objective contributions may be due directly to the decision variables of the segment s_i and $\mathbf{y}_i$, or results of the computations of the segment. Such contributions might include segment duration or maneuvers, which could correspond to minimizing or constraining total tour time or total maneuver magnitudes, for example. Many common objective functions can therefore be decomposed by segment with contributions J_i as

$$J(\mathbf{x}) = \sum_{i=1}^{n_s} J_i(\mathbf{z}_{i-1}, s_i, \mathbf{y}_i) \tag{3.9}$$

and constraints can be decomposed similarly. Thus, in addition to the final states of segments, objective and constraint contributions can be computed and stored per segment.

The ability to decompose and compute tour trajectories by segment allows for the use of constructive approaches. Rather than working only with full-length complete solutions, the search can work with partial solutions and construct them segment-by-segment over many iterations. Further, a constructive approach that also stores the results of segment computations for later re-use can reduce the number of evaluations required for evaluating large numbers of tour trajectories, allowing for exploration of a larger region of the search space.

3.1.2 Tree-Based Solution Representation

Consider a "tree" solution representation. Branches of the tree represent trajectory segments and their decision variables, and nodes of the tree correspond to states at the segment boundaries. A node of the tree may have multiple "children" corresponding to different segments. The tree can grow to arbitrary depth, representing tour trajectories of any length. Results of state, objective and constraint evaluations are stored in the tree, allowing for new solutions to be generated quickly by expanding the tree segment by segment. This allows existing solutions to be used as building blocks for new solutions, with no recomputation of existing segments required. Consider the group of trajectories shown in the left of Figure 3.2. There are eight complete trajectories corresponding to the eight terminal states. Although these trajectories all differ in their final segments, they each share at least one previous segment in common. These trajectories can be represented in a tree as shown at the right of Figure 3.2. New tour trajectories can be created simply by adding additional segments (branches) to nodes in the tree.

The new solution representation is now a single path through the tree. Recall

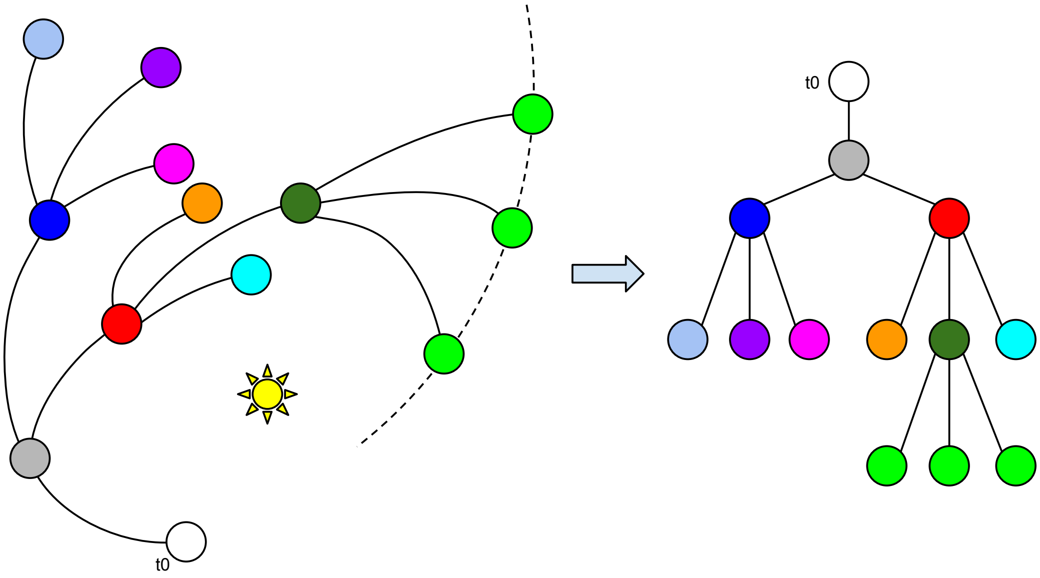


Figure 3.2: A group of trajectories. Colors identify different objects. The final bright green object is visited with three different segments corresponding to different segment decision variables $\mathbf{y}$.

the simple solution representation discussed in Equation (3.1),

$$\mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] \tag{3.10}$$

This same representation can be recovered from a path through the tree. Consider the more detailed view of the tree in Figure 3.2 shown in Figure 3.3. States and decision variables are annotated for two paths, where the superscripts 1 and 2 are used to differentiate them. The nodes are also numbered, where children of nodes are numbered in ascending order starting at 0. Consider the two paths through the tree, $(0, 0, 0, 0)$ and $(0, 0, 1, 1, 2)$. The values of the paths determine which node to move to at each step, beginning at the root of the tree. From these paths, different solutions can be recovered. In terms of the simple solution representation, we can

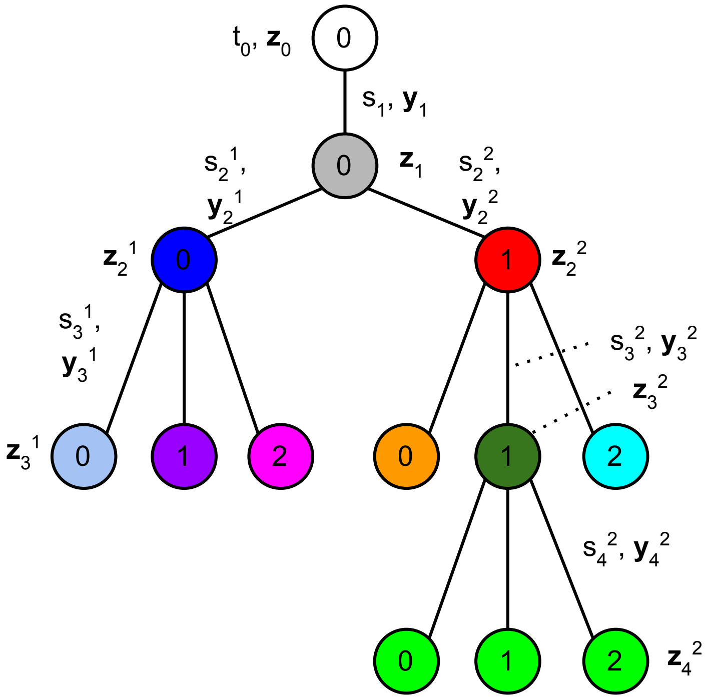


Figure 3.3: A detailed view of the tree representation of Figure 3.2 is shown. Children of nodes are numbered in ascending order starting at 0. Paths can be defined with ordered lists of node numbers starting at the root of the tree. States and decision variables are annotated for two paths, corresponding to the superscripts 1 and 2.

recover from the two example paths the solutions

$$(0,0,0,0) \quad \Longrightarrow \quad \mathbf{x}^1 = \left[t_0, \mathbf{z}_0, s_1, s_2^1, s_3^1, \mathbf{y}_1, \mathbf{y}_2^1, \mathbf{y}_3^1\right] \tag{3.11}$$

$$(0,0,1,1,2) \quad \Longrightarrow \quad \mathbf{x}^2 = \left[t_0, \mathbf{z}_0, s_1, s_2^2, s_3^2, s_4^2, \mathbf{y}_1, \mathbf{y}_2^2, \mathbf{y}_3^2, \mathbf{y}_4^2\right] \tag{3.12}$$

For clarity we will continue to use $\mathbf{x}$ to denote a solution and all of its associated decision variables, understanding that it corresponds to branches and nodes along a path through the tree.

We can analyze the performance of the tree solution representation by considering its use for enumerating all possible solutions for a tour problem. Although a total enumeration is not feasible for even moderately sized problems, this analysis can be used to show the relative performance of the tree solution representation compared to other approaches. Assume that there are N_o candidate objects that can be visited in a tour. Further, assume we compute tours that visit only n_s of these objects, where $n_s \leq N_o$. Each object may be visited only once, and each tour begins from the same initial state. The number of possible tour sequences is then

$$N_{seq} = \frac{N_o!}{(N_o - n_s)!} \tag{3.13}$$

Figure 3.4 shows the possible sequences in a tree solution representation for $N_o = n_s = 5$. The total number of tours possible is much larger than the number of sequences, however, when the continuous decision variables $\mathbf{y}_i$ are allowed to vary for each segment $i \in 1 \ldots n_s$ of each sequence $j \in 1 \ldots N_{seq}$. If we assume that there are K discretizations of the continuous decision variables at each segment, then the number of possible tours is

$$N_{tours} = N_{seq}\, K^{n_s} = \frac{N_o!}{(N_o - n_s)!} K^{n_s} \tag{3.14}$$

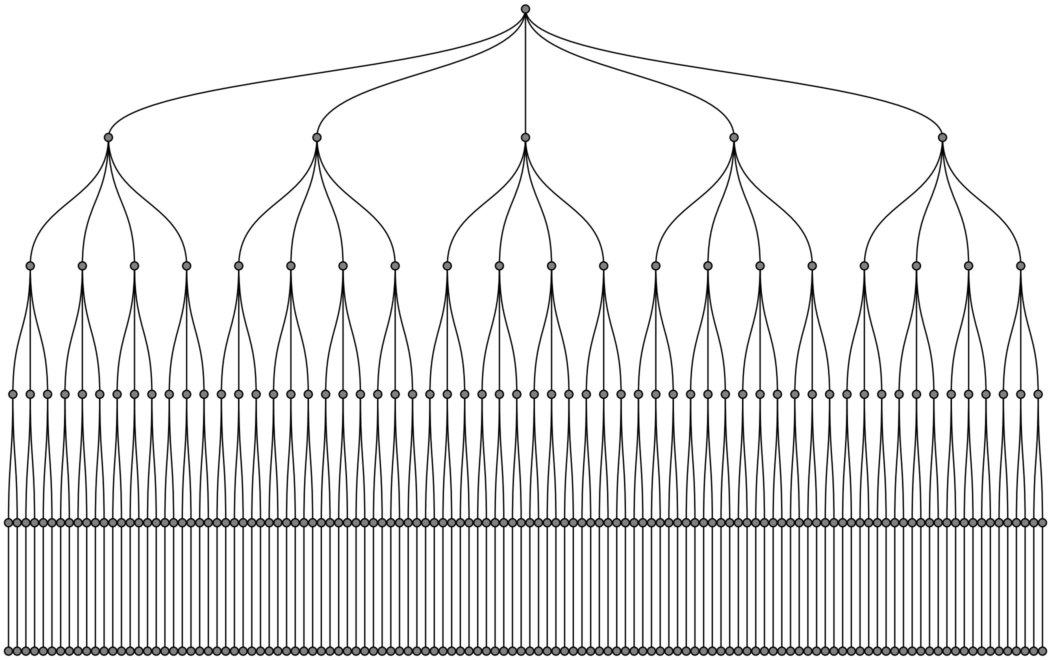


Figure 3.4: All possible tour sequences shown in the tree solution representation for $N_o = 5$ candidate objects and sequence lengths of $n_s = 5$.

We consider the performance in terms of the number of trajectory segment evaluations required, since this is the most computationally expensive operation in the search. For the tree representation, the total number of segment evaluations is

$$\sum_{h=1}^{n_s} \frac{N_o!}{(N_o - h)!} K^h \tag{3.15}$$

If instead each solution is considered independently using another solution representation where segment computations are not stored and reused, then the number of segment evaluations required is larger at

$$n_s \frac{N_o!}{(N_o - n_s)!} K^{n_s} \tag{3.16}$$

We can compare the relative magnitude of these to determine the expected speedup of using the tree solution representation versus enumerating the tours independently, as

$$f = \frac{n_s K^{n_s}}{(N_o - n_s)! \sum\limits_{h=1}^{n_s} \frac{K^h}{(N_o - h)!}} \tag{3.17}$$

We can simplify this expression to find bounds on the expected speedup. If we assume $N_o \to \infty$, $n_s = N_o$ and $K = 1$, we find a lower bound on f to be

$$\underline{f} = \frac{n_s}{e} \tag{3.18}$$

Conversely, if we assume $N_o \to \infty$ and $n_s \to 1$, we find an upperbound on f to be

$$\bar{f} = n_s \tag{3.19}$$

This upper bound corresponds to an ideal linear speedup in the length of the tour.

Figure 3.5 shows the number of tours and speedup for $N_o = 100$ candidate

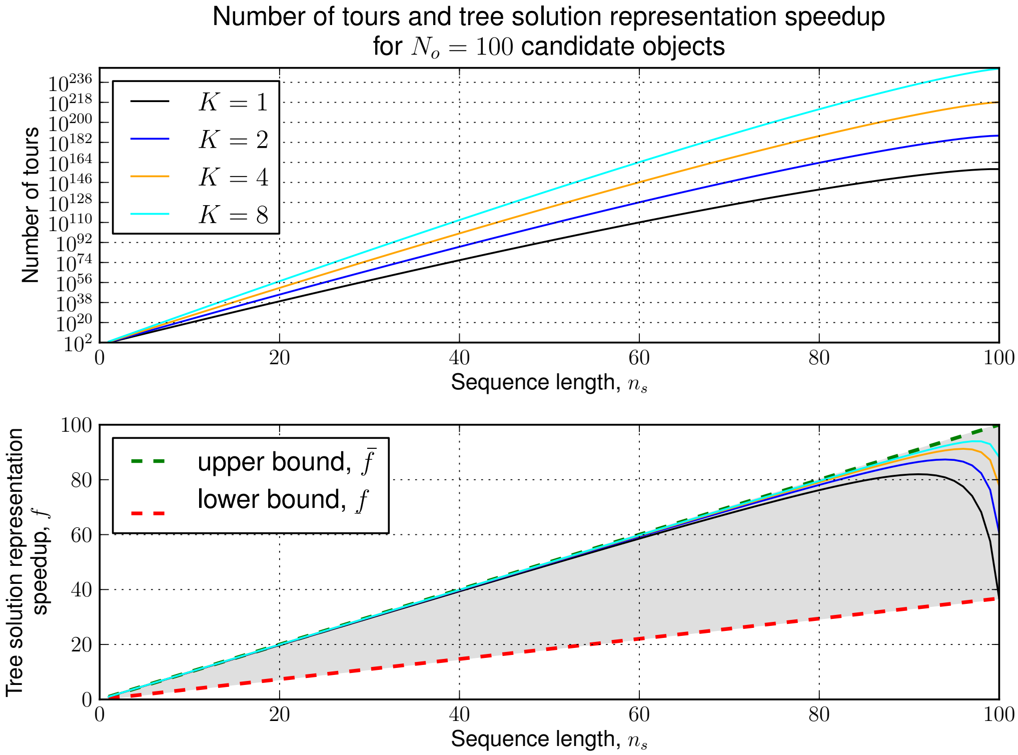


Figure 3.5: Number of tours and tree solution representation speedup for total enumeration of $N_o = 100$ candidate objects and segment discretizations of $K = 1, 2, 4, 8$.

objects and varying tour lengths and segment discretizations. We see a near ideal linear speedup over most of the domain of sequence lengths. This means that the tree solution representation will accelerate the evaluations–and therefore the search–compared to other representations that ignore the hierarchical structure of the search space. Recall, however, that total enumeration is impossible for these problems given the size of the search space; for this small example the number of possible tours quickly exceeds 10^{100}. Any search will only be able to explore a small region of the search space–however, a near linear speedup can be expected over the region of the search space that is explored.

The tree solution representation gives increased computational speed at the cost of storage: the results of all segment computations have to be held in memory. This is in contrast to more typical representations that only require storing one or a fixed number of solutions at a time. For problems with cheap evaluations, such as the traveling salesman, there is less benefit to using the tree representation–the cost of looking up a solution evaluation approaches the cost of simply recomputing it. However, for problems with expensive evaluations, the speedup is significant and is well represented by the current analysis. Finally, we note that the memory required for the tree representation grows as the search progresses. Eventually the memory required will exceed the memory available. In this case, regions of the tree would need to be pruned to make more memory available for the search to continue. This is not an issue for problems with sufficiently expensive evaluations, as the available memory exceeds that required for executing the search over a reasonable period of time.

In summary, we have defined a tree-based solution representation for use in a search. Each solution $\mathbf{x}$ corresponds to a single path through the tree. This solution representation is especially efficient for problems with expensive evaluations: it gives a near linear speedup in the length of the tour. This results from the storage and re-use of segment computations. The representation also facilitates a constructive approach where trajectories are formed segment-by-segment. Additionally, an entire population of solutions is maintained during the search, in contrast to other approaches that maintain only a single (best) solution. The tree solution representation is a building block of a search or optimization algorithm and can be used with many approaches. Aspects of its software implementation are discussed in Appendix A.

3.2 Neighborhoods

Section 3.1 developed a tree-based solution representation for tour trajectories. This solution representation allows for a constructive search approach where partial solutions are expanded segment by segment by adding branches to the tree. However, at this point it has not yet been determined how this expansion of the tree should occur. In practice computing the full tree is infeasible, so it is important to expand it in a way that finds good solutions quickly. This section introduces the concept of neighborhoods, and describes various neighborhoods that can be used to guide the exploration of the tree.

Consider that we have a current solution $\mathbf{x}$, defined as the incumbent solution. We want to determine an improved solution $\mathbf{x}'$ to move to from this incumbent. A neighborhood defines the set of nearby candidate solutions and is denoted as $\mathcal{N}(\mathbf{x})$. Then, as we iteratively move to a new solution $\mathbf{x}'$, we are choosing from available solutions in the neighborhood $\mathcal{N}(\mathbf{x})$. That is,

$$\mathbf{x}' \in \mathcal{N}(\mathbf{x}) \tag{3.20}$$

In a greedy search, we would simply move to the best solution in the neighborhood at every iteration, as

$$\mathbf{x}' = \underset{\mathbf{x} \in \mathcal{N}(\mathbf{x})}{\operatorname{argmin}} \, J(\mathbf{x}) \tag{3.21}$$

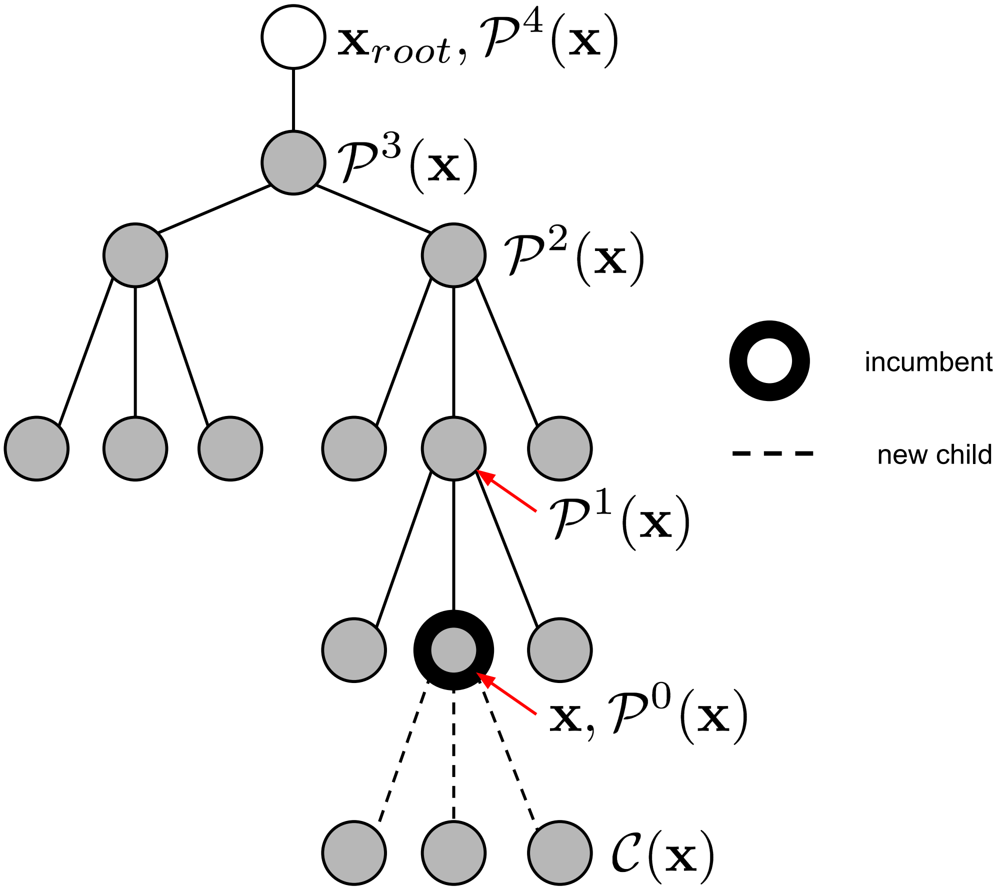
The properties of the neighborhood determine the behavior and performance of the search. For example, one might be tempted to define a very large neighborhood with many candidate solutions in an attempt to find an optimal solution in a small number of iterations. However, since each solution in the neighborhood has to be evaluated to determine which is best, these iterations will be slower. Further, a neighborhood consisting of the entire search space would constitute an exhaustive enumeration of the problem. If this were possible, there would be no need for a search algorithm to begin with. Consequently, neighborhoods are typically defined to only contain a small number of solutions that can be evaluated quickly, and improved solutions are found over many iterations. Neighborhoods can also be defined to intensify or diversify a search about the incumbent solution. Intensification occurs when all candidate solutions in the neighborhood share attributes in common with the incumbent. The search will therefore tend to focus in a smaller region of the search space about the incumbent. Alternatively, a diversifying neighborhood might include solutions sharing little in common with the incumbent, or more explicitly might not have any solutions sharing attributes of the incumbent. The search in this case would tend to quickly move to different regions of the search space. Intensification and diversification can be strategically used to focus on promising regions of the search space or rapidly move away from unfavorable regions.

The remainder of this section develops specific neighborhoods for the tree-based solution representation. These are developed and discussed in terms of their sizes and tendency to diversify or intensify the search.

3.2.1 Neighborhoods for the Tree Solution Representation

Neighborhoods operating on the tree solution representation are constructive neighborhoods since they operate on partial solutions and provide neighboring solutions of longer lengths through expansion of the tree [27]. The tree of solutions grows as neighborhoods are evaluated over many iterations. This is in contrast to most neighborhoods used for combinatorial problems in the operations research literature that only work with complete solutions.

Graph and tree traversal algorithms can be applied and used as neighborhoods in the search given the tree-based solution representation. Before these are considered, let us first define terminology and notation. If we are at an incumbent solution $\mathbf{x}$, let the children of that solution (which are generated by adding segments



$\mathbf{x}_{root}, \mathcal{P}^4(\mathbf{x})$
$\mathcal{P}^3(\mathbf{x})$
$\mathcal{P}^2(\mathbf{x})$
incumbent
new child
$\mathcal{P}^1(\mathbf{x})$
$\mathbf{x}, \mathcal{P}^0(\mathbf{x})$
$\mathcal{C}(\mathbf{x})$

Figure 3.6: An annotated tree is shown for an incumbent solution $\mathbf{x}$. The root of the tree $\mathbf{x}_{root}$, children nodes $\mathcal{C}(\mathbf{x})$ and parent nodes $\mathcal{P}^k(\mathbf{x})$ are labeled.

to the tree) be denoted as $\mathcal{C}(\mathbf{x})$. Methods for generating the solutions $\mathcal{C}(\mathbf{x})$ will be discussed in later sections. A node of the tree is a leaf node if it has no current children. Its immediate children, if any are feasible, are generated and added to the tree when $\mathcal{C}(\mathbf{x})$ is evaluated. The leaf nodes that are descendants of a solution $\mathbf{x}$ are denoted as $\mathcal{L}(\mathbf{x})$. The immediate parent of a solution is $\mathcal{P}^1(\mathbf{x})$. The grandparent is $\mathcal{P}^2(\mathbf{x})$, and further ancestors can be found as $\mathcal{P}^h(\mathbf{x})$. We define $\mathcal{P}^0(\mathbf{x})$ to be the solution $\mathbf{x}$ itself. The root of the tree (corresponding to the initial state) is denoted

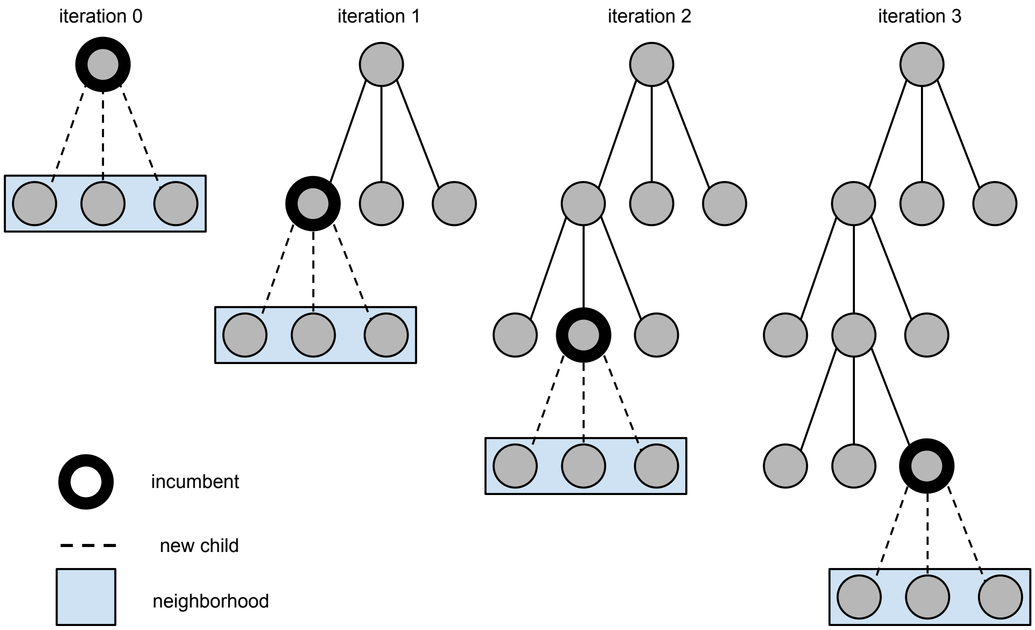


Figure 3.7: The neighborhood definition $\mathcal{N}(\mathbf{x}) = \mathcal{C}(\mathbf{x})$ leads to a depth-first search. At each iteration the incumbent solution moves deeper in the tree.

as $\mathbf{x}_{root}$. Figure 3.6 shows an annotated tree for these definitions.

Depth-First and Breadth-First Search

Consider a partial tour trajectory $\mathbf{x}$ is the current incumbent solution and a leaf node of the tree. Since the incumbent is only a partial solution, there are a set of children solutions that can be constructed by adding segments to it. These children solutions are generated through the evaluation of $\mathcal{C}(\mathbf{x})$. Consider then a neighborhood defined as

$$\mathcal{N}(\mathbf{x}) = \mathcal{C}(\mathbf{x}) \tag{3.22}$$

This neighborhood corresponds to all of the children solutions of $\mathbf{x}$. If at every iteration of the search we move to a solution in this neighborhood, $\mathbf{x}' \in \mathcal{C}(\mathbf{x})$, the result will be a depth-first search as shown in Figure 3.7 [17]. Although a depth-first search will quickly yield tours with long sequences, the resulting solutions are not likely to be near optimal. If we further assume that at each step we move to the best solution (corresponding to the objective definition) in the neighborhood such that

$$\mathbf{x}' = \underset{\mathbf{x} \in \mathcal{C}(\mathbf{x})}{\operatorname{argmin}} \; J(\mathbf{x}) \tag{3.23}$$

then we have a greedy depth-first search. This is analogous to the nearest-neighbor heuristic used in the traveling salesman problem, where at every step the salesman simply moves to the next nearest city until the tour is completed [58]. Such solutions are rarely optimal or near optimal and therefore are typically only used to provide a worst-case bound on the optimal solution. The same is true in this case–the greedy depth-first search will quickly find sub-optimal full length solutions. These solutions, however, can be used as upper bounds on the optimal solution and to establish baseline values for objectives and constraints.

Breadth-first search instead expands all of the children of the incumbent $\mathcal{C}(\mathbf{x})$ before progressing deeper into the tree [17]. The result is that all possible nodes at a given depth of the tree are generated before the search moves deeper in the tree, as shown in Figure 3.8. Thus, all tour trajectories of the shortest length are enumerated before longer tours are generated. Only in the late iterations of the search will full length solutions be found.

The depth-first search neighborhood leads to intensification about the incumbent solution. At every iteration, the neighboring solutions are descendants of the incumbent and therefore possess all attributes of the incumbent: the tour sequence and segment decision variables. Depth-first search neighborhoods can therefore be used to strategically focus on regions of the search space about promising solutions. The breadth-first neighborhood instead leads to diversification since it

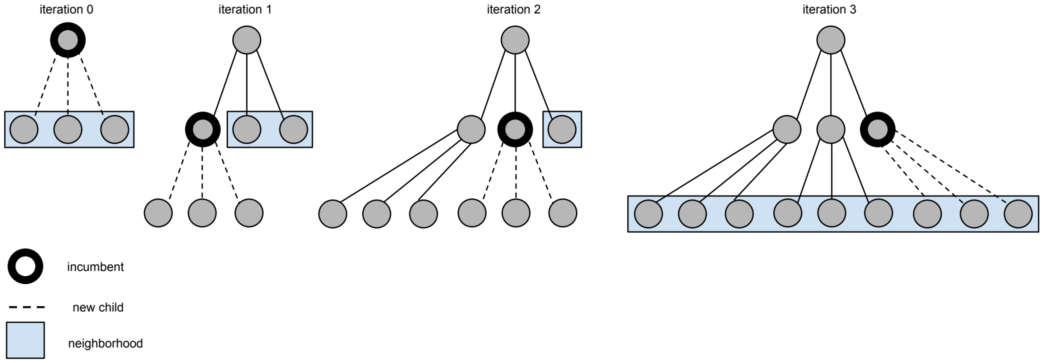


Figure 3.8: The neighborhood corresponds to a breadth-first search. The search explores all nodes at at the highest level of the tree before progressing deeper.

expands nodes highest in the tree first before expanding the incumbent. As the search iterates, it will move to solutions with differing attributes–different sequences and segment decision variable values. A breadth-first neighborhood can be used to strategically move away from the current region of the search space.

Best-First Search

Consider now that instead of expanding the tree with a predefined depth ordering, we explore the most promising nodes of the tree at each iteration. We can define a neighborhood of an incumbent $\mathbf{x}$ as

$$\mathcal{N}(\mathbf{x}) = \mathcal{C}(\mathbf{x}) + \mathcal{L}(\mathbf{x}_{root}) \tag{3.24}$$

where $\mathcal{L}(\mathbf{x}_{root})$ is the set of all leaf nodes in the tree. If at every iteration we move to the new solution $\mathbf{x}'$,

$$\mathbf{x}' = \underset{\mathbf{x} \in \mathcal{C}(\mathbf{x}) + \mathcal{L}(\mathbf{x}_{root})}{\text{argmin}} \quad J(\mathbf{x}) \tag{3.25}$$

then we have a best-first search algorithm [39]. At every iteration, the children of the incumbent solution $\mathcal{C}(\mathbf{x})$ are generated and added to the tree, and the best of

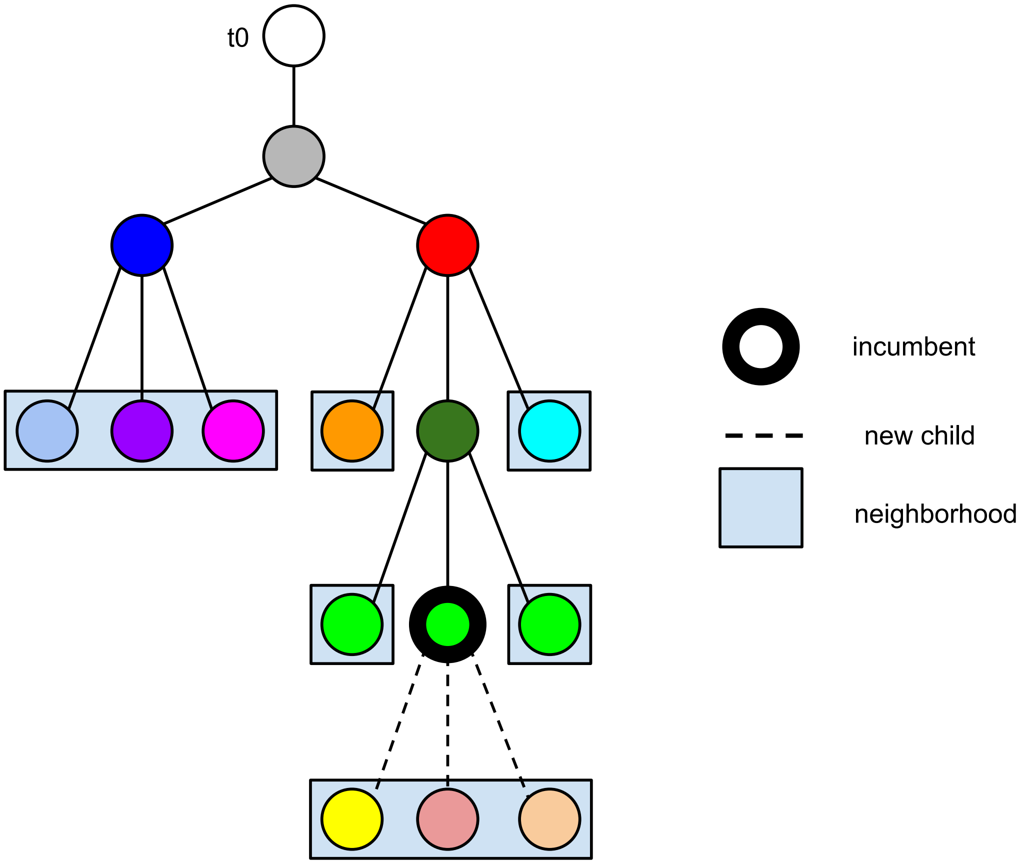


Figure 3.9: A group of tour trajectories represented in the tree solution representation. The best-first neighborhood is highlighted.

all new and existing leaf nodes is moved to. The best-first neighborhood is shown in Figure 3.9.

While depth-first search can be viewed as an intensification strategy, and breadth-first search can be seen as a diversification strategy, best-first search is neither. It instead expands the tree only according to objective values with no preference for solution depth or length. Best-first search is the basis for many graph and tree search algorithms. The primary difference between these algorithms is their definition of the objective. For example, Dijkstra's algorithm is a best-first search using only the objective evaluated on the current partial solution [21]. The A* algorithm instead uses the objective evaluated on the current partial solution in addition to a heuristic term predicting objective contributions of the best complete descendent solution [32]. Best-first search in many cases will find provably optimal solutions. However, a drawback is that as the algorithm iterates, the neighborhood continues to grow as the number of leaf nodes $\mathcal{L}(\mathbf{x}_{root})$ increases. The larger neighborhood is more expensive to evaluate and compare, resulting in slower iterations as the algorithm progresses.

3.2.2 Restricted Best-First Neighborhood

The neighborhood used in best-first search can be modified to allow for strategic intensification or diversification in the search. Recall the best-first search neighborhood is defined as

$$\mathcal{N}(\mathbf{x}) = \mathcal{C}(\mathbf{x}) + \mathcal{L}(\mathbf{x}_{root}) \tag{3.26}$$

An additional parameter h can be added that dictates how far upward in the tree to ascend before finding leaf nodes. While $\mathcal{L}(\mathbf{x}_{root})$ gives all leaf nodes of the tree starting from the root, $\mathcal{L}(\mathcal{P}^h(\mathbf{x}))$ gives all leaf nodes of the tree starting from the h^{th} ancestor of $\mathbf{x}$. The restricted best-first neighborhood is therefore

$$\mathcal{N}(\mathbf{x}, h) = \mathcal{C}(\mathbf{x}) + \mathcal{L}(\mathcal{P}^h(\mathbf{x})) \tag{3.27}$$

This neighborhood is shown in Figure 3.10. Small values of h lead to intensification, with $h = 0$ corresponding to depth-first search, while large values of h allow for diversification. When $h = n_s$, a breadth-first search is possible.

Further, h can be varied to affect the size of the neighborhood. For large trees, large h values can give neighborhoods with many solutions and lengthen the compute time per iteration. While existing leaf nodes do not have to be recomputed

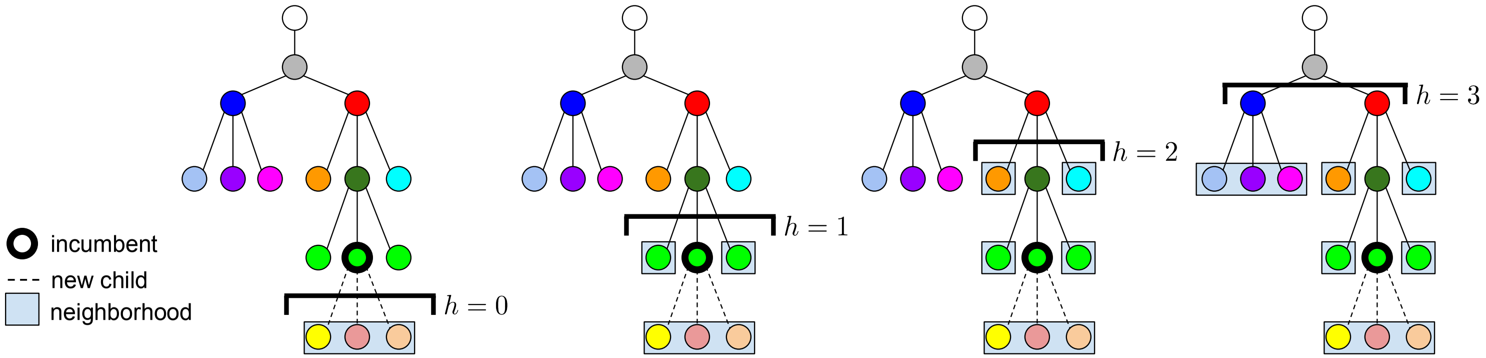


Figure 3.10: A group of tour trajectories represented in the tree solution representation. The restricted best-first neighborhoods corresponding to different h values are shown.

since results are stored in the tree, simply sorting a large number of solutions by objective can be expensive for large neighborhoods–finding the single best solution is $\mathcal{O}(n)$ in the number of solutions [17]. Therefore, in these cases h can be decreased to reduce the size of the neighborhood.

The restricted best-first neighborhood is the most flexible of the tree-based neighborhoods presented so far. It allows for strategic intensification and diversification of the search as well as implicit limiting of the neighborhood size through its parameter h. It will therefore be used as the neighborhood in the overall search algorithm.

3.3 Objectives

The objective $J(\mathbf{x})$ is provided as part of the problem statement. It gives a scalar value that is used to compare solutions to each other and judge which is superior. This objective function is appropriate to use when comparing complete (full-length) solutions. However, in the context of the tree-based solution representation, it instead may be used to compare partial solutions composed of only a few segments with complete solutions composed of many. In these cases, the provided objective function may not be appropriate. For example, objectives based on minimizing resource consumption such as fuel usage will tend to favor partial solutions with few segments. The search would then tend to be a breadth-first search of the tree. Conversely, objectives based on mission goals such as maximizing the number of visited objects will tend to favor complete longer-length solutions. In this case, the search would tend toward a depth-first search. Although the provided $J(\mathbf{x})$ is the metric for judging complete solutions at the end of the search, it should not be used to guide the exploration and construction of partial solutions during the search.

3.3.1 Guiding Objective

A guiding objective can be defined and used during the search in place of the provided objective $J(\mathbf{x})$. The goal of this objective would be to better compare partial solutions of only a few segments with solutions containing many segments. The idea is that a partial solution with a currently poor objective value may produce a better solution if expanded to full length than a near-complete solution with a better objective value. The guiding objective therefore attempts to quantify the potential value of partial solutions. Specifically, it attempts to estimate the best possible solution that can result from expanding a partial solution to full length. In this way, it can more effectively guide the search into promising regions of the search space. This approach is inspired by the objective definition in the A* search algorithm, which has a heuristic term predicting the best achievable objective originating from the current path [55]. While problem-specific heuristics can be developed to determine the value of the guiding objective, this section focuses on developing a general strategy that can be applied to any tour problem and objective definition.

We can define the guiding objective $J^\star(\mathbf{x})$ in terms of a constrained value of the problem, such as a limited mass of fuel or limited mission time. For example, assume the total mission time is limited to t_{max}, according to the tour model of Section 2.2.6. A partial tour begins at time t_0, and each segment ends at time t_i. The final segment of the tour ends at t_{n_s}. The remaining allowed mission time is then $t_{max} - t_{n_s}$. The guiding objective should include an estimate of the best possible complete tour originating from the current partial tour. Assume that the future objective contribution is linear in the time of flight. The contribution can then be defined by a parameter $\frac{dJ^\star}{dt}$. This parameter can be used in the heuristic term in the guiding objective. The guiding objective is then defined as

$$J^\star(\mathbf{x}) = J(\mathbf{x}) + \frac{dJ^\star}{dt}\ (t_{max} - t_{n_s}) \tag{3.28}$$

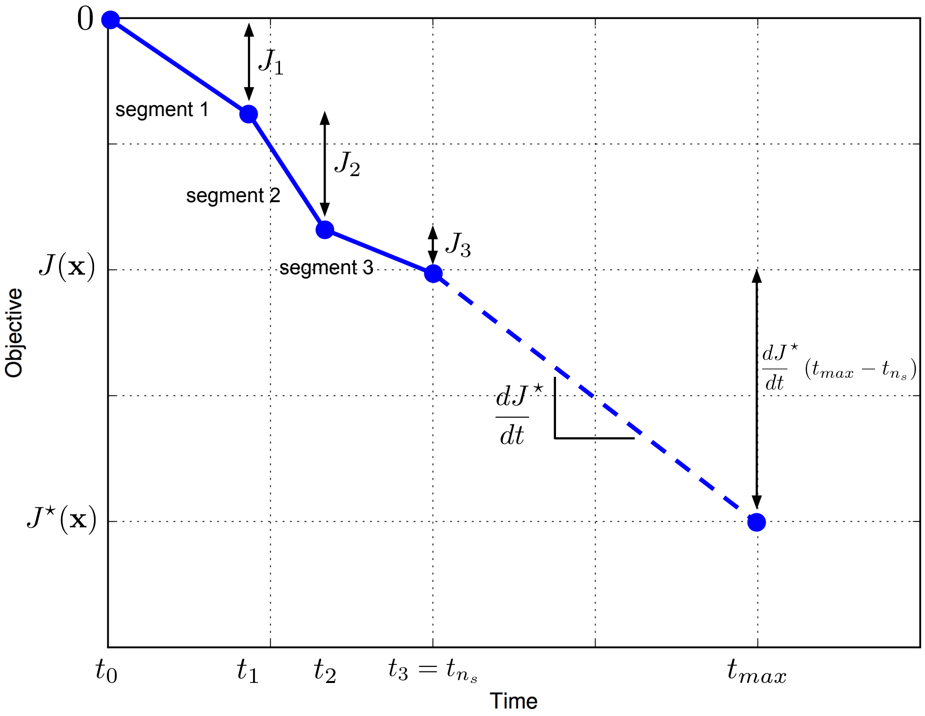


Figure 3.11: The guiding objective contains the objective defined in the problem statement evaluated on the partial trajectory $J(\mathbf{x})$, and a heuristic term based on the parameter $\frac{dJ^\star}{dt}$ estimating contributions over the remaining mission time.

The guiding objective contains the objective defined in the problem statement evaluated on the partial trajectory and a heuristic term estimating contributions over the remaining mission time. Figure 3.11 shows the guiding objective schematically.

The choice of $\frac{dJ^\star}{dt}$ governs the expansion of the tree from iteration to iteration. Ideally the chosen value would perfectly represent the best solution originating from the current solution and therefore guide the search directly to the globally optimal solution. However, since this is not possible, we must consider the effects of under or overestimating the value. For simplicity, assume that the objective we attempt to minimize monotonically decreases as trajectory segments are added to a tour. Such an objective definition would correspond to mission goals: gaining value for every object visited in the tour.

A pessimistic value for $\frac{dJ^\star}{dt}$ will cause the guiding objective to underestimate the best solution possible from a partial solution. Consider the limiting case of $\frac{dJ^\star}{dt} = 0$. This simplifies the guiding objective to the problem-provided objective so that $J^\star(\mathbf{x}) = J(\mathbf{x})$. The guiding objective now predicts no improvement in the objective from adding segments to the partial tour $\mathbf{x}$. Since the objective $J(\mathbf{x})$ monotonically decreases with the tour length, any solution that is expanded will then do better than the guiding objective predicted. The guiding objective will thus tend to favor longer solutions with many segments over shorter solutions with few, and the iterations will tend toward a depth-first search of the tree. The search will quickly yield solutions for long tours.

An optimistic value for $\frac{dJ^\star}{dt}$ will instead cause the guiding objective to overestimate the best solution possible from the current solution. Consider now $\frac{dJ^\star}{dt} = -M$. The guiding objective is then

$$J^\star(\mathbf{x}) = J(\mathbf{x}) - M\ (t_{max} - t_{n_s}) \tag{3.29}$$

If M is sufficiently large, then the guiding objective will predict more improvement than is possible from adding segments to a tour. Any solution that is expanded will do worse than the guiding objective predicted. The guiding objective will thus tend to favor shorter solutions with fewer segments, and the iterations will tend toward a breadth-first search of the tree. We can compare an optimistic guiding objective definition to an “admissible heuristic” in the A* search algorithm [20]. The heuristic is termed admissible since it guarantees that the search will never overlook the possibility of a lower cost path. When a complete solution is found, it must therefore be optimal. The concept extends to this case as well.

In practice we neither want to favor depth-first search nor breadth-first search, but an adaptive exploration of the tree based on the search history. Consider now varying the heuristic term of the guiding objective with

$$0 \geq \frac{dJ^\star}{dt} \geq -M \tag{3.30}$$

If we start the search with $\frac{dJ^\star}{dt} = 0$, then we will quickly find solutions for long tours in a depth-first search manner. We decrease (make more optimistic) $\frac{dJ^\star}{dt}$ as better solutions are found, causing the search to focus on increasingly more promising solutions. Eventually when the magnitude of $\frac{dJ^\star}{dt}$ is large enough, the heuristic term will become admissible and thus guarantee that any complete solutions found will be optimal. The search then has the desirable properties of finding solutions for long tours quickly, continually improved solutions over many iterations and provably optimal solutions as the run time approaches infinity.

$\frac{dJ^\star}{dt}$ can be chosen from a desired goal objective for the problem, J_{goal}. In turn, the goal objective can be adjusted adaptively as the search progresses. Given a value for J_{goal}, we define

$$\frac{dJ^\star}{dt} = \frac{J_{goal}}{(t_{max} - t_0)} \tag{3.31}$$

The value of J_{goal} gives the predicted best solution from the initial conditions of the search. That is,

$$J_{goal} = J^\star(\mathbf{x}_{root}) \tag{3.32}$$

We adjust J_{goal} rather than $\frac{dJ^\star}{dt}$ directly so that the units of the parameter match the units of the problem-provided objective $J(\mathbf{x})$. This allows for easier use of a priori information, such as known achievable objectives, into the search. In the absence of such information, we initialize J_{goal} to zero (thus initializing $\frac{dJ^\star}{dt}$ to zero) and then adaptively decrease it as solutions with objectives better than the goal objective are found. The change in J_{goal} is according to a user-defined parameter,

ΔJ_{goal}. A procedure for the adaptive update is shown in Algorithm 2. We note that the update makes use of only the objective $J(\mathbf{x})$ and not the guiding objective.

Algorithm 2 Adaptively update guiding objective heuristic

```
procedure UpdateGuidingObjectiveHeuristic
    ▷ find the best objective of all solutions in the tree
    $J_{best} \leftarrow \min_{\mathbf{x} \in \mathcal{L}(\mathbf{x}_{root})} J(\mathbf{x})$
    ▷ if the best objective found is better than the goal, then decrease the goal and update $\frac{dJ^\star}{dt}$
    if $J_{best} \leq J_{goal}$ then
        $J_{goal} \leftarrow J_{best} - \Delta J_{goal}$
        $\frac{dJ^\star}{dt} \leftarrow \frac{J_{goal}}{(t_{max} - t_0)}$
        ▷ the guiding objective definition has changed; clear stored $J^\star(\mathbf{x})$ results
        clear stored $J^\star(\mathbf{x})$ results
    end if
end procedure
```

The defined guiding objective $J^\star(\mathbf{x})$ quantifies the potential of a partial solution by estimating the best achievable solution originating from it. This allows for better comparison of solutions of varying lengths. Further, the guiding objective definition is adaptively updated as the search progresses, causing it to favor increasingly promising solutions over time. In the limit as the run time approaches infinity, the guiding objective will lead the search to find provably optimal solutions.

3.3.2 Budget Penalty Terms

Spacecraft tour trajectory problems will all in practice have resource constraints that solutions must satisfy in order to be feasible. Such constraints include a limited mass of available fuel or equivalently a maximum allowed ΔV, for example. These constraints are provided in the problem statement as constraints on the final state of the spacecraft. In the context of our tree-based search, these constraints are evaluated on partial solutions as they are constructed, and solutions violating them are rejected. Assume again that the objective is based on mission goals such as maximizing the number of visited objects. Longer-length solutions will be favored, and we can expect that these longer-length solutions will also be of longer duration. Thus, solutions where all the resources are used very early in the mission are not likely be near optimal. We therefore wish to discourage the search from exploring solutions that use limited resources too quickly. We accomplish this by penalizing the objective when the resource usage exceeds a provided budget.

Consider the impulsive spacecraft tour model defined in Section 2.2.6. If the spacecraft thrusts continuously at a constant magnitude T, then it will deplete its fuel at a constant rate of

$$\dot{m} = -\frac{T}{I_{sp}\, g_0} \tag{3.33}$$

Then, noting that the mass can never fall below the spacecraft's dry mass, its mass at time t can be computed as

$$m(t) = \max\left[m_0 + \dot{m}(t - t_0),\ m_{dry}\right] \tag{3.34}$$

Given the mass history, the corresponding cumulative ΔV can be calculated as

$$\Delta V(t) = I_{sp}\, g_0 \ln \frac{m_0}{m(t)} \tag{3.35}$$

Given a maximum thrust magnitude T_{max}, the maximum allowed mass depletion rate is

$$\dot{m}_{max} = -\frac{T_{max}}{I_{sp}\, g_0} \tag{3.36}$$

The corresponding mass and ΔV profiles define the feasible region for a finite burn trajectory. However, we can also budget the available fuel mass over the maximum allowed duration of the tour so long as we do not exceed this maximum mass rate.

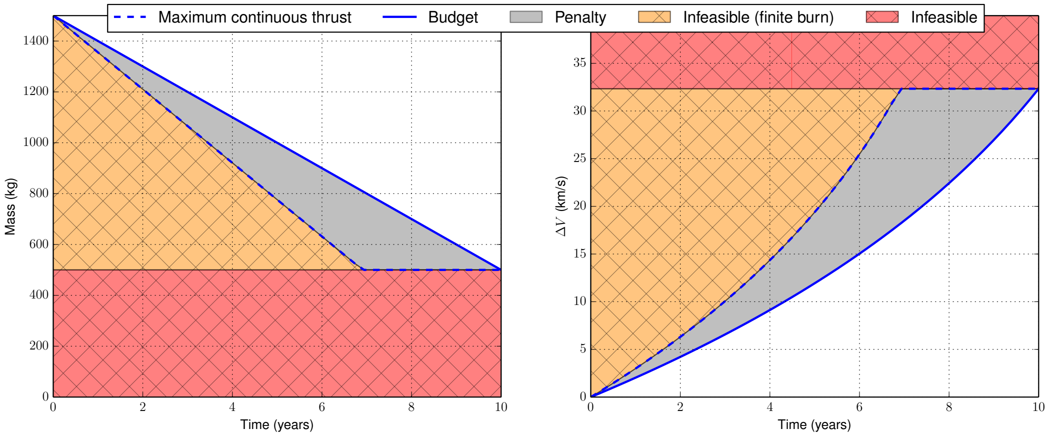


Figure 3.12: The mass and ΔV time histories are shown for maximum continuous thrust until fuel exhaustion and the budgeted amount. The regions where penalties are added to the objective are highlighted. The infeasible regions for a finite burn model and for the final state constraints are shown. The spacecraft parameters correspond to the GTOC4 problem [12].

Then, we have

$$\dot{m}_{budget} = \max\left[-\frac{m_0 - m_{dry}}{t_{max} - t_0},\ \dot{m}_{max}\right] \tag{3.37}$$

The corresponding mass and ΔV profiles define a budget over the time of the trajectory. We wish to penalize trajectories that exceed this budget. Figure 3.12 shows the mass and ΔV histories corresponding to maximum continuous thrust and the budgeted mass usage, and also highlights the regions of infeasibility for both the finite burn and general cases. We penalize solutions falling within the penalty region shown.

Penalty terms can be defined by a non-negative weight and the magnitude exceeding the budget. We can define penalties for both the mass and the ΔV as

$$P_m(\mathbf{x}) = w_m \max\left[m_{budget}(t_{n_s}) - m(t_{n_s}),\ 0\right] \tag{3.38}$$

$$P_{\Delta V}(\mathbf{x}) = w_{\Delta V} \max\left[\Delta V(t_{n_s}) - \Delta V_{budget}(t_{n_s}),\ 0\right] \tag{3.39}$$

Only one such term should have a positive weight, however, as they have similar effects on the objective. These non-negative penalty terms are then added to the guiding objective, giving

$$J^{\star}(\mathbf{x}) = J(\mathbf{x}) + \frac{dJ^{\star}}{dt}\,(t_{max} - t_{n_s}) + P_m(\mathbf{x}) + P_{\Delta V}(\mathbf{x}) \tag{3.40}$$

3.4 Solution Construction (Node Expansion)

This section describes the construction of new solutions during the search. When the search begins, the tree is created with only a root node representing initial conditions. This solution, $\mathbf{x}_{root}$, is the basis for all future solutions generated during the search. These new solutions are generated from iteration to iteration as neighborhoods of incumbents are evaluated. The neighborhood evaluation triggers expansion of nodes in the tree when terms $\mathcal{C}(\mathbf{x})$ representing the children of a solution are evaluated. It is only when these children are explored that new solutions are generated and added to the tree. The remainder of this section details the generation of children solutions $\mathcal{C}(\mathbf{x})$.

When $\mathcal{C}(\mathbf{x})$ is evaluated, trajectory segments are added to the solution $\mathbf{x}$ to generate a set of children solutions. The parent solution's final state is the childrens' initial state. The new children solutions therefore only require the definition of s and $\mathbf{y}$, the next target object and the continuous decision variables of the trajectory segment. The children generated determine the part of the search space that can be explored. Children solutions should therefore be created for all feasible possibilities.

The simplest approach for generating $\mathcal{C}(\mathbf{x})$ would be to enumerate all possible values for s and $\mathbf{y}$, construct the corresponding solutions, and keep the solutions that are feasible to the problem constraints. The enumeration of s is straightforward: consider all values of $s \in O$ feasible to the problem statement. Let us define a function $\mathcal{F}(\mathbf{x})$ that determines the set of feasible objects that may be visited next in a given tour $\mathbf{x}$. If there are no restrictions on the tour itinerary (i.e. each object can be visited any number of times and in any order), then

$$\mathcal{F}(\mathbf{x}) = O \tag{3.41}$$

If, however, there are restrictions on the itinerary, then the feasible objects would instead be a subset of O. If each object may only be visited once in a tour, then

$$\mathcal{F}(\mathbf{x}) = O \setminus \{s_1 \ldots s_{n_s}\} \tag{3.42}$$

We then generate solutions for all $s \in \mathcal{F}(\mathbf{x})$. Then, for each enumeration of s, values for $\mathbf{y}$ must also be enumerated. Since $\mathbf{y}$ represents continuous decision variables, there are an infinite number of possible values. So, a discretization of $\mathbf{y}$ must occur. We define for each element $y_i \in \mathbf{y}$ a range, $y_{i_{min}}$ and $y_{i_{max}}$, and a number of discretizations K_i. If there are n_y elements of $\mathbf{y}$, then there are K total discretizations of the continuous decision variables $\mathbf{y}$, where

$$K = \prod_{i=1}^{n_y} K_i \tag{3.43}$$

The total number of solutions evaluated in the generation of $\mathcal{C}(\mathbf{x})$ is then

$$\|\mathcal{F}(\mathbf{x})\| \, K = \|\mathcal{F}(\mathbf{x})\| \prod_{i=1}^{n_y} K_i \tag{3.44}$$

Only a subset of these solutions will be feasible and define $\mathcal{C}(\mathbf{x})$.

Consider the impulsive tour model of Section 2.2.6 as an example for generating children solutions. In this case, we have

$$\mathbf{y} = \begin{pmatrix} \tau \\ \Delta t \end{pmatrix} \tag{3.45}$$

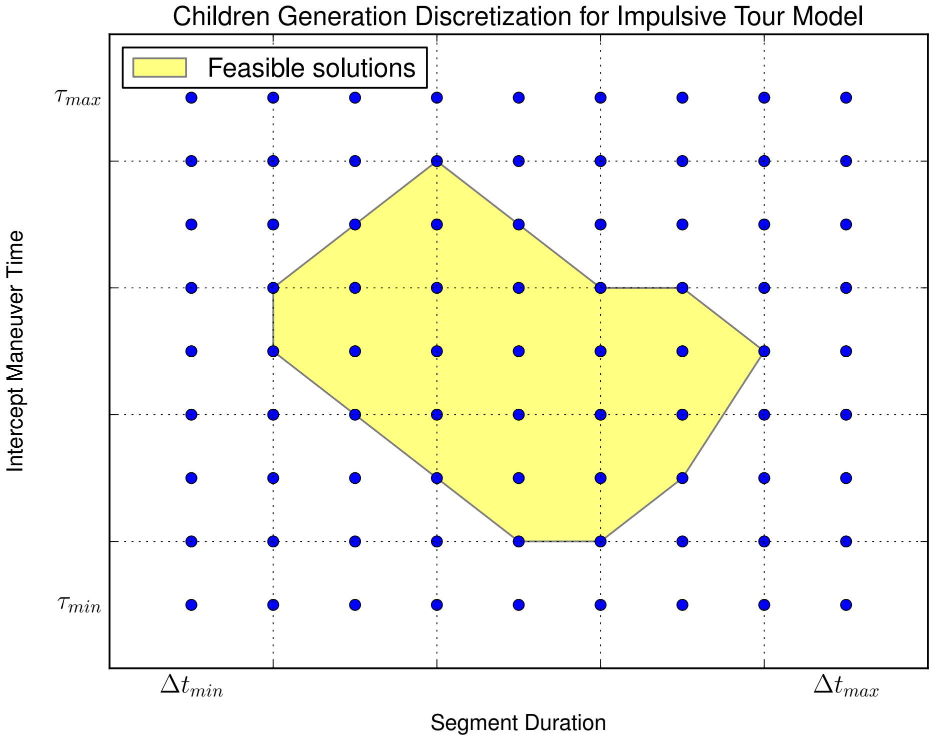


Figure 3.13: For the next target object s, a discretized grid of possible new children solutions over the range of allowed τ and Δt values. The set of solutions found to be feasible after evaluation is highlighted.

We define a corresponding range for the elements as

$$\mathbf{y}_{min} = \begin{pmatrix} \tau_{min} \\ \Delta t_{min} \end{pmatrix} \quad \mathbf{y}_{max} = \begin{pmatrix} \tau_{max} \\ \Delta t_{max} \end{pmatrix} \tag{3.46}$$

and a number of discretizations K_1 for τ and K_2 for Δt. For a single object $s \in \|\mathcal{F}(\mathbf{x})\|$, we can form a grid of $K = K_1 K_2$ possible new children. Figure 3.13 shows the enumeration. Each solution in this grid is evaluated. If the solution is feasible according to the constraints of the problem, then it is added to the set of children solutions $\mathcal{C}(\mathbf{x})$. In this example we see that many more solutions are evaluated than are found feasible. Since these evaluations are computationally expensive, the excess evaluations can slow down the search dramatically. When possible, specific knowledge of the problem should be used to inform this grid search so as to minimize exploration of the infeasible search space.

We have presented a simple method for generating the feasible children solutions $\mathcal{C}(\mathbf{x})$. This method enumerates all possibilities subject to a discretization of the continuous decision variables and discards those solutions that are found to be infeasible. While this approach does generate all the feasible child solutions as desired, it does so inefficiently: many infeasible solutions may be evaluated, and these evaluations are computationally expensive. Chapter 4 presents a space pruning approach that accelerates this approach.

3.5 Tabu Search

Tabu search is a metaheuristic search algorithm first proposed by Glover in 1986 and later formalized in 1989 [24, 25]. Like most metaheuristic approaches, it does not find provably global optimal solutions; it instead is an approximate method that attempts to find good, near-optimal solutions quickly. It is thus widely used on problems where exact methods are not practical. The algorithm is an extension of local search: at every iteration a new solution is moved to from a neighborhood of solutions nearby the incumbent. While local search procedures tend to get stuck at local optima, tabu search explores the search space beyond local optimality and thus is able to find globally good solutions. The fundamental characteristic of tabu search is its use of memory: during the search it records attributes of solutions visited and uses this information to inform future moves and strategy. Unlike other metaheuristic approaches, tabu search does not rely on randomization and is thus deterministic.

Tabu search escapes local optima in two ways. Similar to the “steepest ascent mildest descent" method (for maximization) developed by Hansen, tabu search allows for non-improving moves when the neighborhood has no improving solutions [31]. It moves to the best solution in the neighborhood, which is either the most improving or least disimproving. This allows the algorithm to move away from a local optimum. However, this alone does not prevent the search from returning to the same local optimum on subsequent iterations. This behavior, known as cycling, is prevented in tabu search through the use of memory. During the search, the solutions visited are recorded. These recently visited solutions are then marked as "tabu" such that they cannot be visited again for a given number of iterations. More generally, the search can record attributes of solutions (rather than the solutions themselves) to prevent the search from returning to solutions similar to those already visited. The tabu restrictions act to filter the full neighborhood $\mathcal{N}(\mathbf{x})$ into a candidate neighborhood of admissible (non-tabu) solutions, $\widetilde{\mathcal{N}}(\mathbf{x}) \subseteq \mathcal{N}(\mathbf{x})$. This approach is termed recency-based tabu search. The number of iterations a solution attribute is considered tabu is the tabu tenure. Recency-based memory is one of the most commonly used approaches in tabu search implementations [27].

Advanced tabu search algorithms also employ strategic intensification and diversification [27]. When promising solutions are found, the search can be intensified about them, causing solutions with similar attributes to the promising incumbent to be favored. This results in a more thorough exploration of the promising region of the search space in the hopes that the best solutions in that region will be found. Conversely, diversification may be strategically used when the search is focusing too heavily in a restricted region of the search space. Although good solutions may be found there, more interesting or promising regions of the search space may be left unexplored, and the best solutions to the problem may not be found. Since tabu search is fundamentally a local search procedure, diversification is viewed as a critical component to finding globally good solutions [23]. An extreme form of diversification can be implemented as an escape procedure, causing the search to rapidly move away from the region of the incumbent. Escape procedures are used when simpler forms of diversification have failed. These strategic intensification, diversification and escape methods may be implemented in a variety of ways. Simple approaches involve manipulating the tabu tenure: a short tabu tenure allows for intensification, while a long tabu tenure forces diversification. Other approaches involve changing the definition of the neighborhood through a dynamic neighborhood selection process.

A basic tabu search algorithm demonstrating these approaches is shown in Algorithm 3. The remainder of this section develops components of this tabu search algorithm in terms of the previously developed solution representation, neighborhoods and objectives.

3.5.1 Recency-based Tabu Memory

The fundamental component of the tabu search algorithm described in Algorithm 3 is the recency-based tabu memory used in the isTabu($\mathbf{x}$) and updateTabuAttributes($\mathbf{x}$) procedures. This memory maintains a history of recently visited solutions and prevents the search from returning to those solutions for a period of time, which in turn prevents cycling. The memory can more generally maintain attributes of recently visited solutions, and thus prevent the search from visiting a range of solutions similar to those already visited. The definition of the tabu attributes determines how narrow or broad the resulting tabu restrictions are. We continue by describing the tabu memory structure and defining tabu attributes.

The recency-based tabu memory is stored in an array called the tabu list, which we denote as $\mathcal{T}$. At each iteration, attributes of the new incumbent solution are added to the tabu list. We denote these attributes as $A(\mathbf{x}_i)$ for the incumbent solution of iteration i. Solutions sharing any of the attributes in the tabu list are

Algorithm 3 Basic tabu search algorithm [63].

```
▷ execute tabu search beginning at initial solution $\mathbf{x}_0$
procedure TabuSearch($\mathbf{x}_0$)
    Initialize tabu memory
    ▷ iteration counter
    $i \leftarrow 0$
    repeat
        ▷ find all solutions in the neighborhood
        Evaluate $\mathcal{N}(\mathbf{x}_i)$
        ▷ add non-tabu neighboring solutions to candidate neighborhood
        $\widetilde{\mathcal{N}}(\mathbf{x}_i) \leftarrow \{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i) : \textsc{isTabu}(\mathbf{x}) = \text{False}\}$
        ▷ move to best candidate solution
        $\mathbf{x}_{i+1} \leftarrow \underset{\mathbf{x} \in \widetilde{\mathcal{N}}(\mathbf{x}_i)}{\text{argmin}}\, J(\mathbf{x})$
        ▷ update tabu attributes for the new incumbent solution
        updateTabuAttributes($\mathbf{x}_{i+1}$)
        ▷ strategically intensify or diversify the search if necessary
        if Intensification conditions satisfied then
            intensify($\mathbf{x}_{i+1}$)
        else if Diversification conditions satisfied then
            diversify($\mathbf{x}_{i+1}$)
        end if
        ▷ increment iteration counter
        $i \leftarrow i + 1$
    until Stopping criteria satisfied
    return best solution found
end procedure
```

not admissible in the search and are thus not allowed in the candidate neighborhood $\widetilde{\mathcal{N}}(\mathbf{x})$. The tabu list has a finite length: this is the tabu tenure $N_{\mathcal{T}}$ and a parameter of the search algorithm. A small value of $N_{\mathcal{T}}$ gives a short tabu tenure; solution attributes are then only considered tabu for a short period of time. The result is that the search is allowed to intensify about the incumbent solution. If the tabu tenure is too short, then the search may also cycle about a local optimum. A large value of $N_{\mathcal{T}}$ gives a long tabu tenure and instead has a diversifying effect: the search will tend to move to different regions of the search space. While dynamic adjustment of the tabu tenure is possible, we instead use a static tabu tenure throughout the search. As Glover notes, many problem instances have a robust range of good tabu tenure values that produce good results and can be determined empirically [26]. When the tabu list is full and new attributes are added, the oldest attributes are dropped from the list in a first-in first-out manner and once again allowed in the search. Figure 3.14 shows this behavior.

We now consider tabu attributes for tour trajectory solutions. Recall the solution definition from Equation 3.1,

$$\mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] \tag{3.47}$$

At one extreme we can mark an exact solution as tabu, defining the the tabu attributes of a solution as

$$A(\mathbf{x}) = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] \tag{3.48}$$

This is the narrowest tabu attribute we can apply; it would prevent the particular solution from being visited again for the duration of the tabu tenure. We note however that this type of tabu behavior is implicit in the tree-based solution representation and neighborhood definitions: only leaf nodes are considered in the neighborhood,

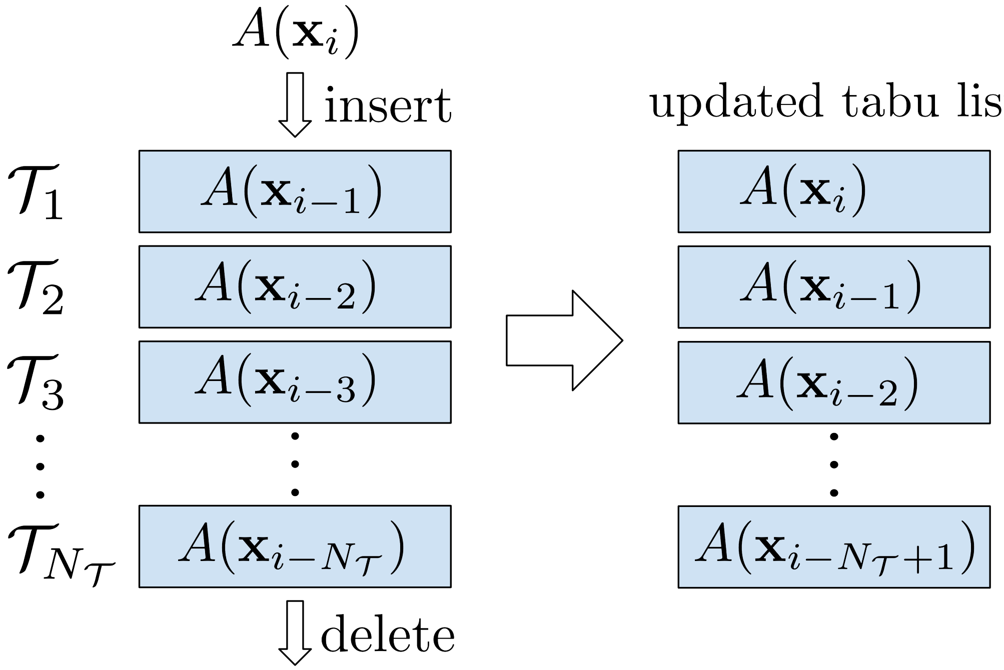


Figure 3.14: The tabu list $\mathcal{T}$ is an array of solution attributes that are prohibited in the search. Its length is the tabu tenure $N_\mathcal{T}$ and determines how many iterations attributes are considered tabu. As attributes of new incumbent solutions $A(\mathbf{x}_i)$ are added to the list, the oldest tabu attributes are forgotten.

and the new incumbent is always expanded at the next iteration. Therefore, an incumbent solution can only ever be visited once, and cycling among exact solutions is impossible. Instead we consider more general attributes of the incumbent. Consider defining a family of trajectories by the ordered sequence of objects they visit. The attribute definition is then

$$A(\mathbf{x}) = [s_1 \dots s_{n_s}] \tag{3.49}$$

This contains only the discrete decision variables of the solution and is a broader tabu restriction on the neighborhood since it matches more solutions. Recall the construction of new solutions during the search described in Section 3.4. From an incumbent solution, new solutions are constructed for each feasible target object for K discretizations of the continuous decision variables $\mathbf{y}$. This results in up to K new feasible tours for the same sequence. More generally, total enumeration from the root of the tree allows up to K^{n_s} solutions for the same sequence. The trajectory family tabu attribute prevents the search from cycling among these similar solutions, promoting the exploration of varying sequences even if one particular sequence is promising.

We can formalize the procedure for testing if a solution is tabu and therefore admissible or not in the candidate neighborhood. This procedure $\textsc{isTabu}(\mathbf{x})$ is referenced in the basic tabu search algorithm (Algorithm 3) and is defined below in Algorithm 4. If any of the attributes of the solution are present in the tabu list, then the solution is tabu. After a non-tabu solution is chosen as the new incumbent, the tabu list is updated with attributes of the new solution. This procedure, updateTabuAttributes($\mathbf{x}$) adds tabu attributes of $\mathbf{x}$ to the tabu list and removes the oldest tabu attributes. Algorithm 5 shows the procedure. The definition of tabu attributes $A(\mathbf{x})$, the tabu list $\mathcal{T}$ and the tabu tenure $N_{\mathcal{T}}$ define all that is necessary for the implementation of recency-based tabu memory.

Algorithm 4 Test if a solution $\mathbf{x}$ is tabu.

```
▷ test if solution $\mathbf{x}$ is tabu
procedure isTabu($\mathbf{x}$)
    if $A(\mathbf{x}) \in \mathcal{T}$ then
        return True
    else
        return False
    end if
end procedure
```

Algorithm 5 Update tabu attributes.

```
▷ update tabu attributes for new incumbent solution $\mathbf{x}$
procedure updateTabuAttributes($\mathbf{x}$)
    ▷ shift all tabu attributes back in the tabu list
    ▷ the last tabu attribute is removed
    for all $i \in 1 \ldots N_{\mathcal{T}} - 1$ do
        $\mathcal{T}_{i+1} \leftarrow \mathcal{T}_i$
    end for
    ▷ add attributes of solution $\mathbf{x}$ to the front of the tabu list
    $\mathcal{T}_1 \leftarrow A(\mathbf{x})$
end procedure
```

3.5.2 Strategic Intensification and Diversification

We have defined the recency-based tabu memory with the primary goal of preventing cycling. This short-term memory can be viewed as a basic form of diversification: it prevents the search from visiting similar solutions in sequence within the limits of the tabu tenure. We now consider methods to more strategically guide the search beyond the basic recency-based approach. Specifically, we consider an approach that adaptively intensifies and diversifies the search based on its performance. When we intensify the search, we choose to move toward solutions similar to the incumbent. Conversely, diversification guides the search away from such solutions. The remainder of this section develops the conditions that trigger intensification and diversification as well as the modifications to the search that implement them.

The conditions that trigger intensification and diversification can be based on any component of the search. For example, we can choose to intensify the search when a new best solution is found. The resulting intensification would favor solutions sharing attributes of that best solution in the hopes of finding more improved solutions. We can also diversify the search when solutions are visited too often in order to explore a more representative sample of the search space; this approach is typically implemented with a longer term frequency-based tabu memory [27]. In determining these triggering conditions, however, we must consider the unique characteristics of our search and our solution representation in particular. The tree-based solution representation represents partial solutions, and the neighborhoods we have defined expand or contract these partial solutions in constructive and destructive processes. From iteration to iteration, the incumbent solution thus varies in length, ranging from partial solutions with few trajectory segments to complete solutions with many. We wish to maintain a balanced exploration during the search, both visiting long-length complete solutions that solve the original problem statement and expanding short-length partial solutions to reveal new promising paths. This is another form of diversity in the search, not based on specific attributes of solutions but instead on their length. The length of a solution is defined by its sequence length n_s or equivalently its depth in the tree. In the context of our search then, we define intensification to cause solutions to be expanded to longer lengths. These longer-length solutions share trajectory segments in common with the incumbent, or may even be completely based on the incumbent. Diversification instead moves upward in the tree to shorter solutions that share little in common with the incumbent.

We define a stall condition for identifying when the search should be intensified or diversified. A stall occurs when the search is stuck in a limited range of solution lengths over a specified number of iterations. The stall condition is thus identified with two parameters: a minimum solution length range $\underline{R}$ and a period of iterations over which the range is measured N_{stall}. The range of solution lengths over the previous N_{stall} iterations is given by

$$R(N_{stall}) = \max_{\mathbf{x} \in \left\{\mathbf{x}_i \ldots \mathbf{x}_{i-N_{stall}}\right\}} \mathbf{x}\left[n_s\right] - \min_{\mathbf{x} \in \left\{\mathbf{x}_i \ldots \mathbf{x}_{i-N_{stall}}\right\}} \mathbf{x}\left[n_s\right] \tag{3.50}$$

A stall thus occurs when

$$R(N_{stall}) \leq \underline{R} \tag{3.51}$$

Figure 3.15 shows an example search history where the search proceeds depth-first and then stalls. After a period of iterations the stall ends and the search proceeds deeper in the tree. The resulting solution length range calculations are shown. The stall condition is not evaluated until at least N_{stall} iterations have occurred. We see the stall is detected N_{stall} iterations after it begins.

Once a stall is detected, we wish for the search to dynamically adjust in order to break the stall. Based on our development so far, we have two options: we can either intensify the search about the incumbent or instead diversify. Either approach can break the stall. An intensification can force the search deeper in the tree to

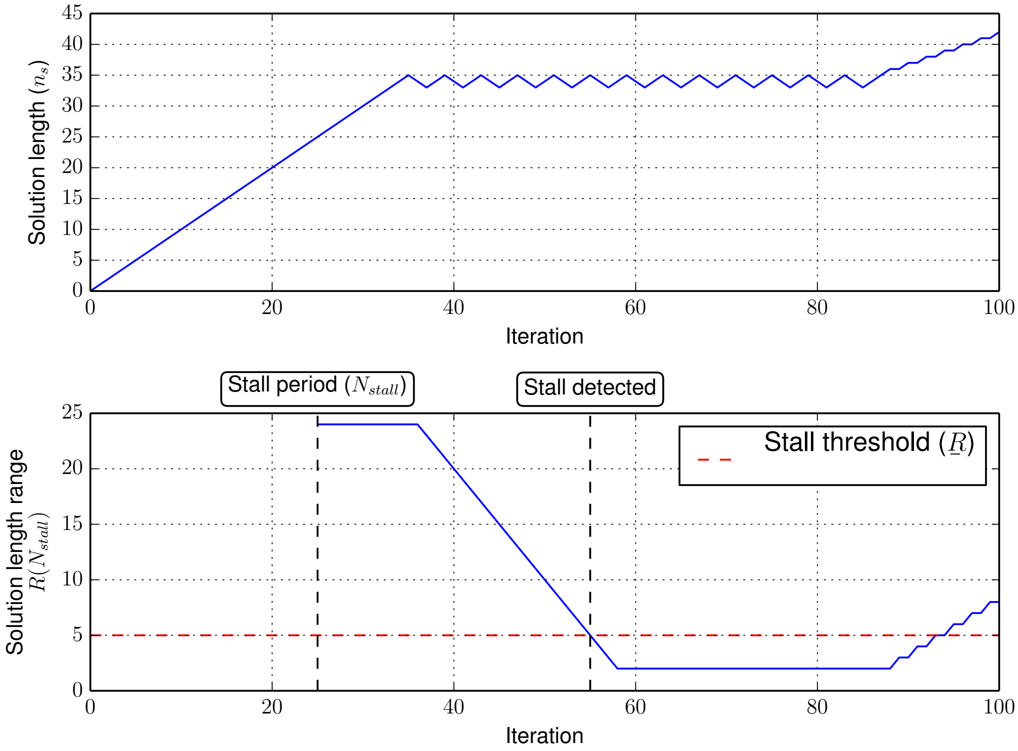


Figure 3.15: An example search history. The incumbent solution lengths are shown along with the stall condition evaluated for $N_{stall} = 25$ and $\underline{R} = 5$.

longer solution lengths, while a diversification will allow the search to move upward in the tree to shorter solutions. We thus choose to react to stalls using both of these approaches, using the properties of the incumbent to determine which approach is used. In order to optimize the problem-provided objective $J(\mathbf{x})$, we generally want to find long-length solutions. We therefore choose by default to react to stalls with intensification to encourage these long-length solutions. However, intensification in the form we have described works only when the incumbent solution can be feasibly expanded. When a tour is near the limits of the problem constraints, any expansion of the tour will lead to infeasible children solutions. At this point, intensification can no longer break a stall and will instead prolong it. When this occurs, we instead choose to break the stall with diversification, which allows the search to move to shorter-length solutions and find new promising paths in different regions of the search space. The condition at which we choose diversification rather than intensification may be defined in terms of any of the problem constraints, such as limited fuel mass or mission duration. We have already developed budget constraints to limit fuel usage in Section 3.3.2, so we choose the maximum mission duration as the limiting condition. Specifically, when the incumbent solution's mission duration is sufficiently close to the limit, we choose to react with diversification. When a stall is detected we react with intensification or diversification according to the conditions below.

$$\text{stall reaction} = \begin{cases} \text{intensification} & t_{n_s} - t_0 \leq 0.9\ (t_{max} - t_0) \\ \text{diversification} & t_{n_s} - t_0 > 0.9\ (t_{max} - t_0) \end{cases} \tag{3.52}$$

We can now detect stalls and decide whether to intensify or the diversify the search in order to escape them. We now need to determine how the desired intensification and diversification should be implemented. For this we can dynamically change the neighborhood. Dynamic neighborhood selection is an advanced approach used on its own in metaheuristic local search procedures as well as in tabu search implementations [44, 27]. Recall the restricted best-first neighborhood developed in Section 3.2.2, defined as

$$\mathcal{N}(\mathbf{x}, h) = \mathcal{C}(\mathbf{x}) + \mathcal{L}(\mathcal{P}^h(\mathbf{x})) \tag{3.53}$$

The only parameter of the neighborhood is h which dictates how far upward to move in the tree before finding leaf nodes. We found that small values of h led to intensification, with $h = 0$ forcing a depth-first search. Large values of h allow for diversification up to the limit of breadth-first search. We now choose to adaptively update h to achieve intensification or diversification–decreasing it to intensify and increasing it to diversify. We define additional parameters to bound the values of h, such that

$$h_{min} \leq h \leq h_{max} \tag{3.54}$$

Algorithms 6 and 7 define procedures for intensifying and diversifying the search. These procedures are called when the stall condition is satisfied according to the conditions in Equation (3.52).

Algorithm 6 Intensify the search about the solution **x**.

```
procedure INTENSIFY($\mathbf{x}$)
    ▷ intensify by updating the restricted best-first neighborhood $\mathcal{N}(\mathbf{x}, h)$
    ▷ subject to $h_{min} \leq h \leq h_{max}$
    if $h > h_{min}$ then
        $h \leftarrow h - 1$
    end if
end procedure
```

While the intensification and diversification approaches developed are sufficient for addressing most stalls, there may still be scenarios where a stall persists. For these cases we develop an escape procedure to more drastically react. Escape

Algorithm 7 Diversify the search about the solution **x**.

```
procedure DIVERSIFY(x)
    ▷ diversify by updating the restricted best-first neighborhood $\mathcal{N}(\mathbf{x}, h)$
    ▷ subject to $h_{min} \le h \le h_{max}$
    if $h < h_{max}$ then
        $h \leftarrow h + 1$
    end if
end procedure
```

procedures are another component of tabu search that are used in reactive tabu search methods and ejection chain approaches [27]. We implement the escape procedure as a stronger form of the diversification in Algorithm 7. That approach adjusted the h parameter of the neighborhood by an increment of 1 on each call, up to a maximum for h_{max}. In the escape procedure we instead increment h by $\Delta h_{escape} \ge 1$ with no maximum limit. The escape procedure is shown in Algorithm 8. The conditions for triggering the escape procedure are expectedly more strict. First, we only consider using the escape procedure when diversification has failed to break a stall. Since the diversification procedure may be called multiple times and increase the neighborhood parameter h up to a value of h_{max}, we only consider triggering the escape procedure when $h = h_{max}$. Further, we only consider escaping if the stall has been active for $N_{escape} \ge N_{stall}$ iterations. The triggering conditions for the escape procedure are thus

Algorithm 8 Escape from the region of the search space near the solution **x**.

```
procedure ESCAPE(x)
    ▷ diversify by updating the restricted best-first neighborhood $\mathcal{N}(\mathbf{x}, h)$
    ▷ no constraints on $h$
    $h \leftarrow h + \Delta h_{escape}$
end procedure
```

$$h \ge h_{max} \ \wedge \ R(N_{escape}) \le \underline{R} \tag{3.55}$$

At this point we have developed methods for reacting to stalls, including intensification, diversification and in extreme cases an escape procedure. Each of these approaches changes the neighborhood in order to break the stall. When the search is operating nominally–that is, exploring a sufficiently diverse range of solution lengths–we should revert the neighborhood to a nominal state. We thus define a value of the solution length range $\bar{R} > \underline{R}$ indicative of a sufficiently diverse exploration. When the range of solution lengths meets or exceeds this range, we react by relaxing or diversifying the neighborhood up to the limit of h_{max} using the diversify procedure already defined. The search is already exploring sufficiently diverse solution lengths in this case; the diversification increases the size of the neighborhood to allow for more promising paths to be found. The triggering conditions for the relaxation are then

$$R(N_{stall}) \geq \bar{R} \tag{3.56}$$

The strategic intensification, diversification and escape procedures–as well as the neighborhood relaxation–can now all be integrated into a single dynamic neighborhood selection strategy. However, we introduce one final parameter: a cool down period, $N_{cooldown}$. The cool down period gives a minimum number of iterations between changes to the neighborhood, restricting how often neighborhood changes may occur. This in turn allows the effects of a neighborhood change to be measured before more changes are made. Algorithm 9 shows the combined neighborhood modification procedure.

3.6 Algorithm Summary

The previous sections developed the components of the overall global search methodology. These include the tree-based solution representation for efficiently handling tour trajectory solutions and neighborhoods that operate on that solution repre-

Algorithm 9 Dynamically update the neighborhood based on the search performance.

```
procedure UPDATENEIGHBORHOOD($\mathbf{x}_i \ldots \mathbf{x}_0$)
    ▷ do nothing if we are in the cool down period
    if $i - neighborhoodUpdated \leq N_{cooldown}$ then
        return
    end if
    ▷ evaluate the escape conditions
    if $R(N_{escape}) \leq \underline{R} \wedge h \geq h_{max}$ then
        ESCAPE($\mathbf{x}_i$)
        $neighborhoodUpdated \leftarrow i$
        return
    end if
    ▷ evaluate the stall condition
    if $R(N_{stall}) \leq \underline{R}$ then
        ▷ the iterations have stalled
        ▷ intensify or diversify to break the stall depending on solution duration
        if $\mathbf{x}_i[t_{n_s}] - \mathbf{x}_i[t_0] \leq 0.9\ (t_{max} - \mathbf{x}_i[t_0])$ then
            INTENSIFY($\mathbf{x}_i$)
        else if $\mathbf{x}_i[t_{n_s}] - \mathbf{x}_i[t_0] > 0.9\ (t_{max} - \mathbf{x}_i[t_0])$ then
            DIVERSIFY($\mathbf{x}_i$)
        end if
        $neighborhoodUpdated \leftarrow i$
        return
    end if
    ▷ increase the size of the neighborhood if the search is proceeding nominally
    if $R(N_{stall}) \geq \bar{R}$ then
        DIVERSIFY($\mathbf{x}_i$)
        $neighborhoodUpdated \leftarrow i$
        return
    end if
end procedure
```

sentation to enable a local search approach. A guiding objective was defined for use during the search to better compare solutions of varying lengths. The process of constructing new solutions based on existing partial solutions was outlined. Finally, a tabu search algorithm was developed in terms of these building blocks. The tabu search algorithm includes a recency-based tabu memory to avoid cycling as well as a dynamic neighborhood selection procedure to strategically intensify and diversify the search for better performance. Table 3.1 summarizes the components and parameters of the search. Algorithm 10 gives the complete algorithm.

The use of the tree-based solution representation is novel in the context of tabu search. It allows a constructive approach that is especially beneficial for problems with expensive solution evaluations. Further, the tree solution representation implicitly maintains a population of solutions, thereby providing a collection of solution alternatives at the termination of the search rather than just a single best solution.

Name	Description
$\mathbf{x}_i$	Incumbent solution of iteration i; $\mathbf{x}_0$ is the initial solution
$[y_{i_{min}}, y_{i_{max}}]$	Range of allowed values for the i^{th} element of the continuous segment decision variables $\mathbf{y}$
K_i	Number of discretizations for the i^{th} element of the continuous segment decision variables $\mathbf{y}$
$\mathcal{T}$	Tabu list
$N_{\mathcal{T}}$	Tabu tenure
$A(\mathbf{x})$	attributes of solution $\mathbf{x}$ to consider for tabu status
$\mathcal{N}(\mathbf{x}, h)$	Restricted best-first neighborhood of solution $\mathbf{x}$
h	Restricted best-first neighborhood: maximum ascendance in tree
$[h_{min}, h_{max}]$	Range of allowed values for h for intensification or diversification
Δh_{escape}	Change in h for escape procedure
$\widetilde{\mathcal{N}}(\mathbf{x})$	Candidate neighborhood of non-tabu solutions
$J^{\star}(\mathbf{x})$	the guiding objective for use during the search (different than the problem-provided objective)
$\frac{dJ^{\star}}{dt}$	parameter of the guiding objective that predicts objective contributions over the remaining allowed time
ΔJ_{goal}	parameter for updating $\frac{dJ^{\star}}{dt}$ when a new best solution is found
$w_m, w_{\Delta V}$	budget penalty weights for mass and ΔV terms used in the guiding objective
$N_{cooldown}$	Cool down period: number of iterations between allowed changes to the neighborhood
N_{stall}	Stall period: number of iterations over which the stall condition is evaluated
N_{escape}	Escape period: number of iterations over which the escape condition is evaluated
$\underline{R}$	Stall threshold: a stall occurs when the range of solution lengths is below $\underline{R}$ for N_{stall} iterations
$\bar{R}$	Relaxation threshold: when the range of solution lengths is above $\bar{R}$, the neighborhood is relaxed

Table 3.1: Summary of components and parameters for the tabu search algorithm.

Algorithm 10 Tabu search algorithm.

```
▷ execute tabu search beginning at initial solution $\mathbf{x}_0$
procedure TabuSearch($\mathbf{x}_0$)
    ▷ initialize tabu memory
    $\mathcal{T} \leftarrow \varnothing$
    ▷ iteration counter
    $i \leftarrow 0$
    repeat
        ▷ find all solutions in the neighborhood
        Evaluate $\mathcal{N}(\mathbf{x}_i, h)$
        ▷ add non-tabu neighboring solutions to candidate neighborhood
        $\widetilde{\mathcal{N}}(\mathbf{x}_i) \leftarrow \{\mathbf{x} \in \mathcal{N}(\mathbf{x}_i, h) : \textsc{isTabu}(\mathbf{x}) = \text{False}\}$
        ▷ move to best candidate solution according to the guiding objective
        $\mathbf{x}_{i+1} \leftarrow \underset{\mathbf{x} \in \widetilde{\mathcal{N}}(\mathbf{x}_i)}{\text{argmin}}\, J^\star(\mathbf{x})$
        ▷ update tabu attributes for the new incumbent solution
        updateTabuAttributes($\mathbf{x}_{i+1}$)
        ▷ dynamically change the neighborhood based on the search history
        ▷ includes strategic intensification, diversification and escape procedures
        updateNeighborhood($\mathbf{x}_{i+1} \ldots \mathbf{x}_0$)
        ▷ increment iteration counter
        $i \leftarrow i + 1$
    until Stopping criteria satisfied
    return best solution found
end procedure
```

Chapter 4

Search Space Pruning

In this chapter we develop a search space pruning approach to accelerate the construction of new feasible trajectories. The goal of pruning procedures in general is to prune out regions of the search space known to be infeasible or non-optimal. This reduces the size of the search space and accelerates the underlying optimization algorithm. In the context of our work, we apply search space pruning to the solution construction (node expansion) phase of the tree-based tabu search algorithm developed in Chapter 3. We continue by describing the brute-force approach already developed and its performance characteristics. We then develop the search space pruning procedure and compare its performance to the brute-force approach.

4.1 Brute-force Approach

Section 3.4 developed a basic procedure for solution construction. Recall that at each iteration, new trajectories are constructed by adding new trajectory segments to an incumbent solution $\mathbf{x}$. Although many trajectory segments may be computed, only those that are found to be feasible to the problem constraints are kept. These feasible trajectory segments are then added to the tree-based solution representation, and

$[\tau_{min}, \tau_{max}]$	Range of allowed segment intercept maneuver times (fraction of segment duration)
K_τ	Discretizations of τ
$[\Delta t_{min}, \Delta t_{max}]$	Range of allowed segment durations
$K_{\Delta t}$	Discretizations of Δt

Table 4.1: Sampling parameters for trajectory segments in the augmented impulsive tour model.

those found to be infeasible are discarded after computation. The basic approach developed was a total enumeration of all possible segment decision variable values subject to bounds and discretization. Recall that $\mathcal{F}(\mathbf{x})$ gives us the set of target objects that may be feasibly visited next in a solution $\mathbf{x}$, and that we make K_i discretizations of each of the n_y continuous decision variable for $i \in 1 \ldots n_y$. A brute force approach enumerates all of these possibilities. The number of trajectory segments created during expansion of a single node is thus

$$\|\mathcal{F}(\mathbf{x})\| \prod_{i=1}^{n_y} K_i \tag{4.1}$$

However, the total number of these trajectory segments that are feasible can be much less depending on the problem constraints.

We now consider the specific case of the augmented impulsive tour model developed in Section 2.2.6 for the rest of the development. We again have $\mathcal{F}(\mathbf{x})$ representing the feasible next target objects. However now we have two continuous decision variables,

$$\mathbf{y}_i = \begin{pmatrix} \tau_i \\ \Delta t_i \end{pmatrix} \tag{4.2}$$

representing the time of the intercept maneuver and the duration of the trajectory segment i. These are subject to minimum and maximum bounds and a discretization that are summarized in Table 4.1. The number of trajectory segments computed in the brute force approach for this model is now

$$\|\mathcal{F}(\mathbf{x})\| \, K_\tau K_{\Delta t} \tag{4.3}$$

The definition of $\mathcal{F}(\mathbf{x})$ is problem-specific, but an upper bound is the full set of N_O target objects. Therefore, an upper bound on the number of trajectory segment computations for expansion of a single node in the tree for any problem is

$$N_O K_\tau K_{\Delta t} \tag{4.4}$$

We can further quantify the computations based on the number of Kepler propagations and Lambert targeting calls required, which we denote as k and l, respectively. The Lambert targeting procedure is in general more computationally expensive than a Kepler propagation; the specific performance difference depends on the underlying algorithms used for each and their implementation. The evaluation of a trajectory segment in this case requires two Kepler propagations. The first finds the position of the spacecraft at the time of the intercept maneuver,

$$t_{i-1} + \tau_i \Delta t_i \tag{4.5}$$

The second computes the position of the target object at the end time of the segment,

$$t_{i-1} + \Delta t_i \tag{4.6}$$

When $\tau_i = 0$ then the first Kepler call is not required as the final state of the previous segment is known; however in the general case we must account for it. A single Lambert targeting call then computes the maneuver between these two positions over the elapsed time. Thus, for the brute force approach expanding the solution $\mathbf{x}$ and reusing the results of computations when allowed we have

$$k_{BF} = K_\tau K_{\Delta t} + \|\mathcal{F}(\mathbf{x})\| K_{\Delta t} \tag{4.7}$$

$$l_{BF} = \|\mathcal{F}(\mathbf{x})\| K_\tau K_{\Delta t} \tag{4.8}$$

We wish to minimize both of these quantities in a more efficient solution construction procedure. Since the Lambert targeting procedure is more expensive, we wish to especially minimize its use. We will measure the performance of the space pruning procedure we develop in terms of the number of required computations k and l.

4.2 Trajectory Envelopes

This section describes the use of trajectory envelopes to prune infeasible trajectory segments from the search space. The trajectory envelope forms the reachable domain of the spacecraft; any target objects not intersecting this envelope cannot be feasibly visited, and therefore should not be considered in solution construction. This reduces the number of trajectory segment computations required which in turns accelerates the search. However, computing the trajectory envelope has its own associated cost as well.

Vinh et al. introduced the concept of the reachable domain, which is a set of points attainable by an interceptor with a limited ΔV capability [68]. Their work analyzed the reachable domain for interception of targets at hyperbolic speeds for short times of flight. They proved that the largest reachable domain is achieved when the maximum allowed impulse is applied as early as possible (under the assumption of short time of flight). Independent works by Wen and Xue develop analytical approaches for computing the reachable domain of a spacecraft in elliptic motion subject to a single upper-bounded impulsive maneuver [70, 73]. However, the approaches compute the reachable domain without consideration to time: either the limited trajectory duration or the time accessibility of the reachable space (e.g. a target object's orbit may intersect the reachable domain but may not be reachable at the time of intersection). We develop an alternative simulation-based approach that defines the reachable domain both based on position in space and time.

We begin by generating a collection of spacecraft trajectories for a set of possible impulsive maneuvers. Each trajectory begins at the final state of the previous segment $\mathbf{z}_{i-1}(t-1)$ where

$$\mathbf{z}_{i-1}(t_{i-1}) = \begin{pmatrix} \mathbf{r}(t_{i-1}) \\ \mathbf{v}(t_{i-1}) \\ m(t_{i-1}) \end{pmatrix} \tag{4.9}$$

For the purpose of computing the trajectory envelope, we assume that maneuvers occur at the beginning of the trajectory segment with the largest allowed magnitude. We can compute the maximum feasible magnitude of an impulsive maneuver based on the mass of the spacecraft and the maximum allowed trajectory segment duration. We further assume that the impulsive maneuver is an idealization for a low-thrust spacecraft, and constrain the impulse appropriately. The maximum thrust magnitude T_{max} and specific impulse I_{sp} are provided as part of the spacecraft parameters given in Chapter 2 (Table 2.2). Assuming the spacecraft thrusts over the maximum duration of a trajectory segment, then the maximum corresponding impulsive maneuver magnitude is

$$\Delta V_{max} = I_{sp}\, g_0 \ln \left(\frac{m(t_{i-1})}{m(t_{i-1}) - \frac{T_{max}}{I_{sp} g_0} \Delta t_{max}} \right) \tag{4.10}$$

Figure 4.1 shows values for ΔV_{max} for different allowed trajectory segment durations and the spacecraft parameters of the fourth annual global trajectory optimization competition (GTOC4) shown in Table 5.2. The resulting values depend also on the

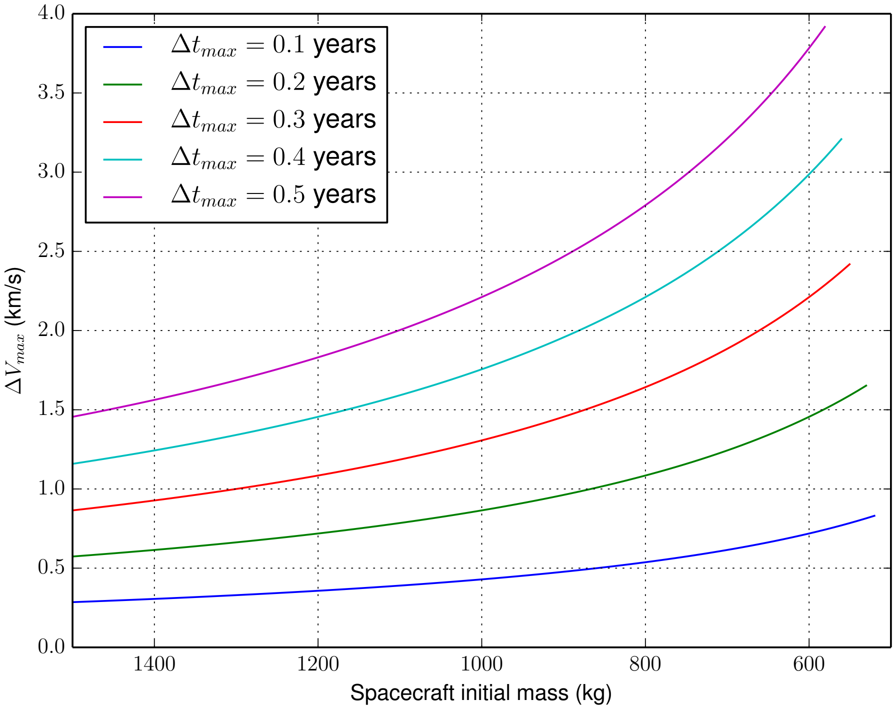


Figure 4.1: The values for ΔV_{max} are shown over varying spacecraft initial mass and values of Δt_{max}. The spacecraft parameters correspond to the fourth annual global trajectory optimization competition (GTOC4) summarized in Table 5.2.

initial mass of the spacecraft at the trajectory segment–the spacecraft becomes more efficient as its mass decreases. Using this approach we can compute an appropriate ΔV_{max} when new trajectory segments are to be generated during solution expansion. We note that a range of trajectory segment durations are allowed; we compute ΔV_{max} corresponding to Δt_{max} which is an upper bound for values associated with lesser values of Δt.

The impulsive maneuver may occur in any unit direction, which we can parameterize with spherical angles α and β as

$$\hat{\mathbf{u}} = \begin{pmatrix} \cos\alpha\cos\beta \\ \sin\alpha\cos\beta \\ \sin\beta \end{pmatrix} \tag{4.11}$$

Then, the set of maximum allowed impulsive maneuvers is given by

$$\Delta\mathbf{V}_{max} = \Delta V_{max}\ \hat{\mathbf{u}}(\alpha, \beta) \tag{4.12}$$

for any values of α and β. We can discretize the angles to form a finite set of these maneuvers; we allow α to vary between $[0, 2\pi]$ and β to vary between $[-\pi/2, \pi/2]$. Within these ranges we choose K_α and K_β evenly spaced values for α and β respectively. Then, the total number of maneuver discretizations is

$$K_{\Delta V} = K_\alpha K_\beta \tag{4.13}$$

Figure 4.2 shows the discretizations of $\Delta\mathbf{V}_{max}$. The set of possible maneuvers form a sphere of radius ΔV_{max}.

We can now enumerate the $K_{\Delta V}$ possible maneuvers to form the set of initial conditions for the trajectories forming the trajectory envelope. These initial

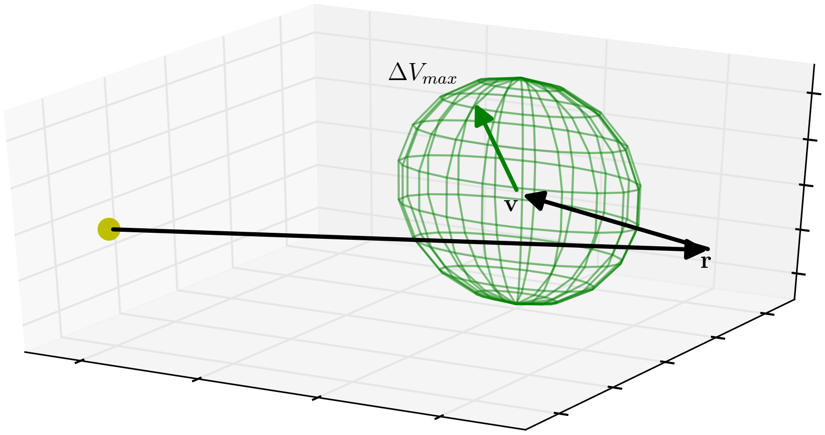


Figure 4.2: Discretization of all possible impulsive maneuvers for a magnitude of ΔV_{max}. The maneuver direction is discretized over the spherical angles α and β for a total of $K_{\Delta V} = K_\alpha K_\beta$ discretizations.

conditions are given by

$$\mathbf{X}_j(t_{i-1}) = \begin{pmatrix} \mathbf{r}(t_{i-1}) \\ \mathbf{v}(t_{i-1}) + \Delta V_{max}\ \hat{\mathbf{u}}(\alpha_j, \beta_j) \end{pmatrix} \quad \text{for } j = 1 \ldots K_{\Delta V} \tag{4.14}$$

We can propagate each of these $K_{\Delta V}$ states forward in time by Δt_{max}. The resulting trajectories form the trajectory envelope that bounds the reachable domain of the spacecraft given a single maneuver originating at the state $\mathbf{z}_{i-1}(t_{i-1})$ with an upper-bounded maneuver magnitude of ΔV_{max}. Figure 4.3 shows an example of the trajectories forming the envelope.

A target object whose orbit intersects this envelope may be reachable by the spacecraft; otherwise it can be pruned from the search space. Intersecting the envelope is a necessary but not sufficient condition for reachability. First, ΔV_{max} is an upper bound computed for the maximum segment duration Δt_{max}. Secondly, intersecting the spatial region of the envelope does not mean the target object is

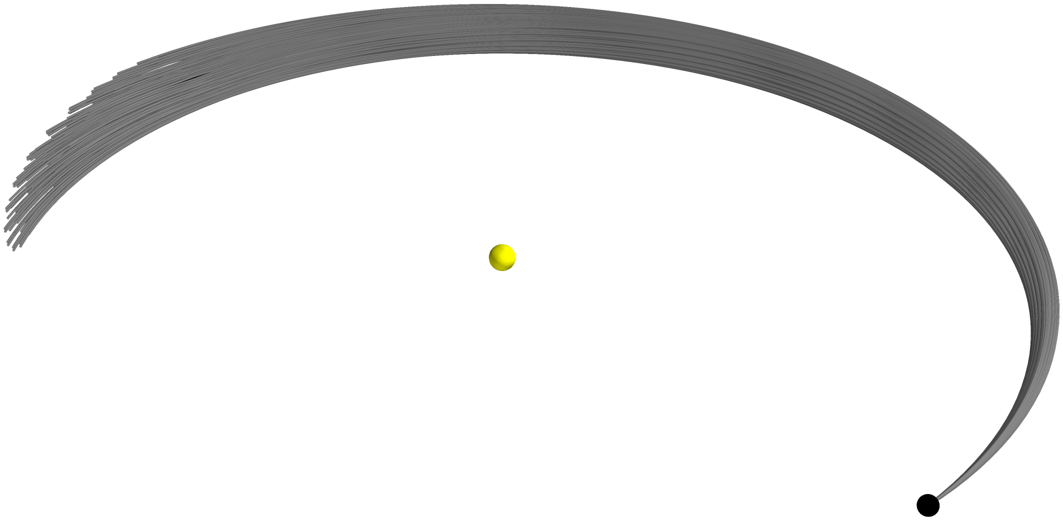


Figure 4.3: The trajectories forming a trajectory envelope are shown for $\Delta V_{max} =$ 1 km/s and $\Delta t_{max} = 0.55$ years. The spherical angles α and β are each discretized over a 15 degree spacing.

reachable at the time of intersection; the trajectory envelope is again an upper bound in this regard. Finally, although we can visualize this envelope, we do not yet have a method for determining when such target object orbit intersections occur.

4.2.1 Bounding Boxes

This section develops a procedure for determining when a target object's orbit intersects the trajectory envelope. At a high level we need to determine if a path-volume intersection occurs, where the path is the target object orbit and the volume is the trajectory envelope. However, although we have the trajectories bounding the envelope, we do not have an analytical representation of its volume. Computing these intersections is thus challenging. It is analogous to a collision detection problem, where it must be determined if two objects of arbitrary shape intersect or not. Such problems commonly occur in physical simulations, for example. A common approach to accelerate this collision detection is to generate simpler shapes that bound the more complicated shape of the object. If the simpler shapes intersect, then the objects themselves may collide–otherwise no collision is possible. The most common approach is the use of bounding boxes; for each object a bounding box is computed that contains the shape. The bounding box is an upper bound on the actual volume. The problem we face here is most similar to the ray tracing algorithm of computer graphics, where individual rays of light are traced through a scene, and ray-object intersections must be computed to determine visibility and lighting [56]. We apply and extend the bounding box procedure commonly used in such algorithms to our case here.

The trajectory envelope bounds a three-dimensional region of space. However, the trajectories making up the envelope also include time information. For example, the later parts of the envelope are only reachable by the spacecraft for longer times of flight (approaching Δt_{max}). Even if a target object intersects the spatial volume of the trajectory envelope then, it may still not be reachable due to the timing of the intersection. We therefore add another dimension to our bounding boxes–time–and create multiple bounding boxes over discrete time intervals. A path now intersects a bounding box if it intersects its spatial volume at a time within its time bounds. The bounding box is still only a necessary condition for reachability, but it represents a tighter upper bound with the additional timing restriction. A bounding box may be oriented arbitrarily; however for simplicity we use axis-aligned bounding boxes such that the spatial bounds are along the Cartesian dimensions of the underlying coordinate system. Therefore, each bounding box is represented by the spatial and time bounds shown in Figure 4.4.

We create N_{BB} bounding boxes over the feasible trajectory segment durations $[\Delta t_{min}, \Delta t_{max}]$ provided in the trajectory segment sampling parameters. The

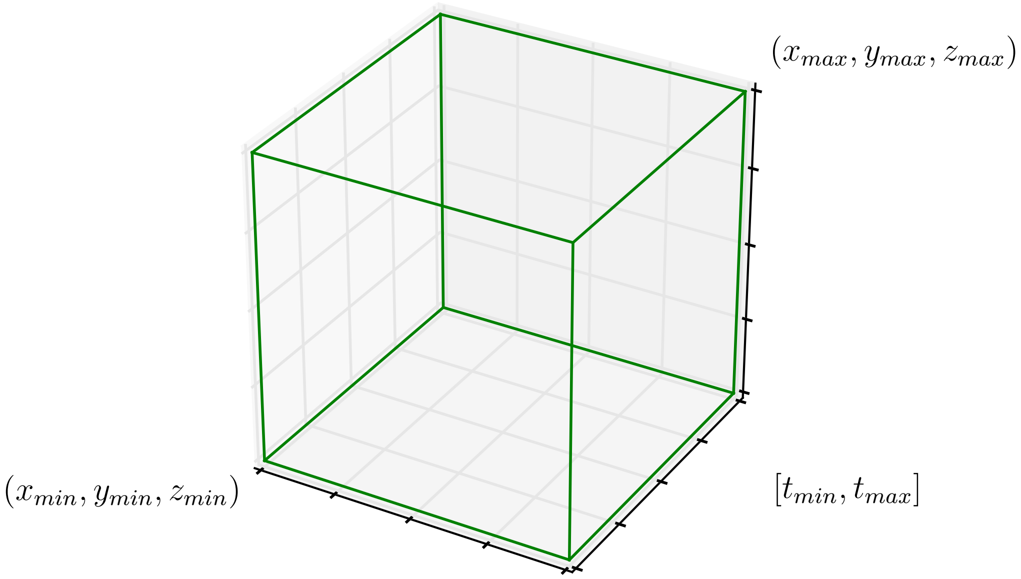


Figure 4.4: An axis-aligned bounding box bounds a volume in the spatial region of ($[x_{min}, x_{max}]$, $[y_{min}, y_{max}]$, $[z_{min}, z_{max}]$). We additionally constrain the bounding box with a time range $[t_{min}, t_{max}]$.

minimum and maximum time bounds for each bounding box b are then given by

$$\begin{aligned} (t_{min})_b &= t_{i-1} + \Delta t_{min} + \frac{b-1}{N_{BB}} \left(\Delta t_{max} - \Delta t_{min}\right) \\ (t_{max})_b &= t_{i-1} + \Delta t_{min} + \frac{b}{N_{BB}} \left(\Delta t_{max} - \Delta t_{min}\right) \end{aligned} \quad \text{for } b = 1 \ldots N_{BB} \tag{4.15}$$

These time bounds are constant once the bounding box is created. The spatial extents of a bounding box are formed by propagating the trajectories that form the envelope (Equation (4.14)) and expanding the bounding boxes over discrete points of each trajectory. A bounding box is expanded according to Algorithm 11. Similarly,

Algorithm 11 Expand the bounding box b given the position (x, y, z) at time t.

```
procedure expandBoundingBox($b$, $(x, y, z)$, $t$)
    ▷ the current spatial extents are $([x_{min}, x_{max}], [y_{min}, y_{max}], [z_{min}, z_{max}])_b$
    ▷ expand the bounding box only if within its time bounds
    if $(t_{min})_b \leq t \leq (t_{max})_b$ then
        $(x_{min})_b \leftarrow \min[x, (x_{min})_b]$
        $(y_{min})_b \leftarrow \min[y, (y_{min})_b]$
        $(z_{min})_b \leftarrow \min[z, (z_{min})_b]$

        $(x_{max})_b \leftarrow \max[x, (x_{max})_b]$
        $(y_{max})_b \leftarrow \max[y, (y_{max})_b]$
        $(z_{max})_b \leftarrow \max[z, (z_{max})_b]$
    end if
end procedure
```

we can test if a trajectory intersects a bounding box by testing if any point along its trajectory is within its spatial and time bounds; Algorithm 12 shows the procedure. These two procedures form the basis for the overall search space pruning procedure.

We form the spatial extents of all the bounding boxes by propagating each of the $K_{\Delta V}$ initial states of Equation (4.14) forward in time and expanding each bounding box as required. We consider K_{BB} discrete time intervals for each of the N_{BB} bounding boxes. Therefore, we have $N_{BB}K_{BB} + 1$ samples of each trajectory

Algorithm 12 Test if the bounding box b contains the position (x, y, z) at time t.

```
procedure intersectsBoundingBox($b$, $(x, y, z)$, $t$)
    ▷ the current spatial extents are $([x_{min}, x_{max}], [y_{min}, y_{max}], [z_{min}, z_{max}])$
    ▷ test the position only if within bounding box time bounds
    if $(t_{min})_b \leq t \leq (t_{max})_b$ then
        if $(x_{min})_b \leq x \leq (x_{max})_b$ and $(y_{min})_b \leq y \leq (y_{max})_b$ and $(z_{min})_b \leq z \leq (z_{max})_b$ then
            return True
        end if
    end if

    return False
end procedure
```

and $K_{\Delta V}(N_{BB}K_{BB} + 1)$ total samples for the trajectory envelope. Each of these samples requires a single Kepler propagation. Figure 4.5 shows $N_{BB} = 10$ bounding boxes computed for the trajectory envelope of Figure 4.3. We see that the bounding boxes overlap in space, and that the overlap increases toward the end of the envelope. This is expected as the trajectories forming the envelope diverge from each other over time.

Finally, given the bounding boxes associated with the trajectory envelope, we can compute when target objects intersections with the envelope may occur (in space and time). As in the brute force approach, we sample each target object in the feasible set $\|\mathcal{F}(\mathbf{x})\|$ over $K_{\Delta t}$ times. In each case we propagate the target object to the time $t_{i-1} + \Delta t_i$ and test for a bounding box intersection. Figure 4.6 shows an example of the target object bounding box intersections for the GTOC4 problem. If the target object intersects a bounding box at a given time, then we construct trajectory segments to the target object at that time and compute the required maneuvers using the Lambert procedure. If we assume the number of target object bounding box intersections is a factor of f less than the total number enumerated, then the complexity of this approach in terms of Kepler propagations k and Lambert

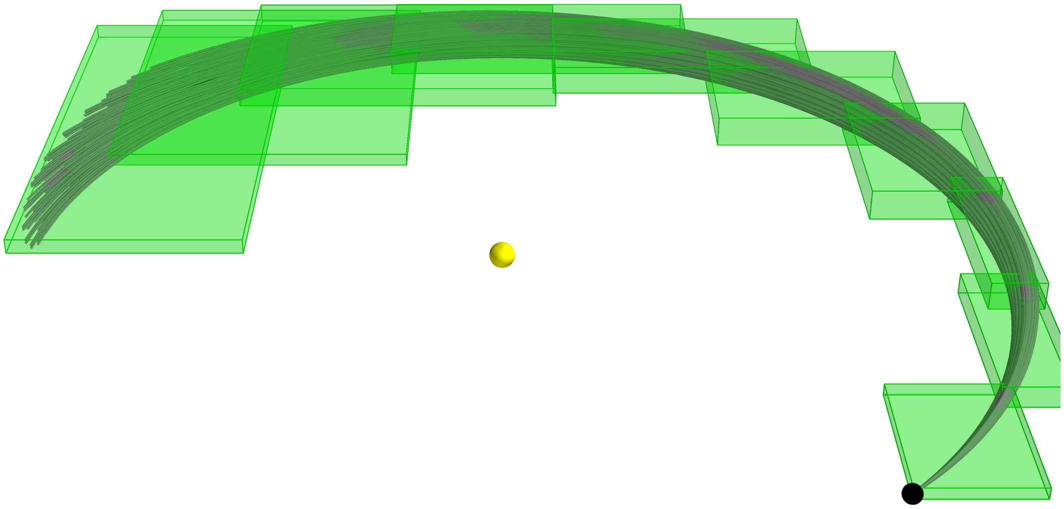


Figure 4.5: $N_{BB} = 10$ bounding boxes are shown for the trajectory envelope of Figure 4.3 ($\Delta V_{max} = 1$ km/s).

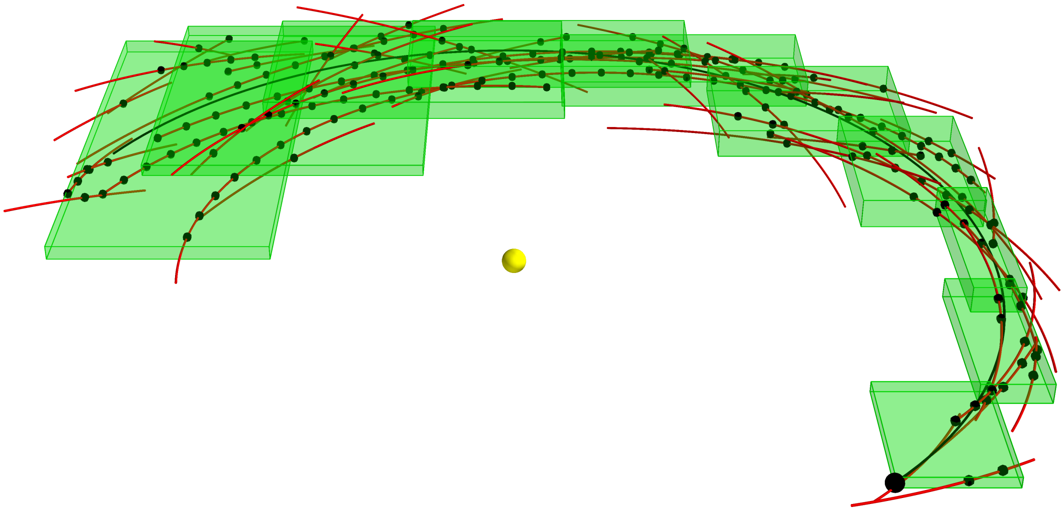


Figure 4.6: The target object bounding box intersections are shown for the case in Figure 4.5.

targeting calls l is

$$k_{SP} = K_{\Delta V}\left(N_{BB}K_{BB} + 1\right) + \left\|\mathcal{F}(\mathbf{x})\right\| K_{\Delta t} + K_{\tau}K_{\Delta t} \tag{4.16}$$

$$l_{SP} = \frac{\left\|\mathcal{F}(\mathbf{x})\right\| K_{\tau}K_{\Delta t}}{f} \tag{4.17}$$

The first term of Equation (4.16) represents the computations that expand the bounding boxes of the trajectory envelope; the second gives the required propagations to compute target object bounding box intersections; and the third is an upper bound on computing the position of the spacecraft at the time of maneuver for the required Lambert targeting calls of Equation (4.17). We see an increase in the required Kepler propagations of $K_{\Delta V}\left(N_{BB}K_{BB} + 1\right)$ due to the bounding box generation, but a decrease by a factor of f in required Lambert targeting calls. Since the Lambert procedure is more computationally expensive, we expect to see a performance increase. Further, the increase in k is of constant complexity, while the decrease in l scales with the number of target objects, and is thus of greater benefit for larger problems with many target objects.

4.2.2 Summary

The search space pruning approach we have developed forms a trajectory envelope bounding the reachable domain of the spacecraft. The trajectory envelope is based on an upper-bounded single impulsive maneuver occurring at the beginning of a trajectory segment. In order to more simply compute intersections of target object orbits with this envelope and to consider the time feasibility of resulting trajectory segments, we developed an approach based on bounding boxes. The approach results in fewer trajectory segments being created, and therefore reduces the number of Lambert procedure calls and accelerates solution construction. The trade-off is an increase in Kepler propagations required to generate the envelope. The parameters

ΔV_{max}	Maximum allowed impulsive maneuver
K_α, K_β	Number of discretizations of the spherical angles α and β for possible impulsive maneuver directions
$K_{\Delta V}$	Total number of discretizations for impulsive maneuvers (equal to $K_\alpha K_\beta$)
N_{BB}	Number of bounding boxes
K_{BB}	Number of time intervals for each bounding box

Table 4.2: Parameters for the trajectory envelope search space pruning procedure.

of the search space pruning approach are shown in Table 4.2. We further note that since the procedure is simulation-based, it can easily be applied to dynamics other than two-body motion.

4.3 Performance

This section considers the relative performance of the search space pruning approach compared to the brute force method. Here we only consider the number of required computations: Kepler propagations k and Lambert targeting calls l. We consider the full performance impact of the space pruning approach later for the GTOC4 problem in Section 5.3.3.

We fix the sampling parameters for the search space pruning procedure at $N_{BB} = 10$ bounding boxes and $K_{BB} = 10$ time intervals per bounding box. We discretize the spherical angles of the unit impulse direction at intervals of 15 degrees, resulting in $K_\alpha = 24$ and $K_\beta = 13$ for a total of $K_{\Delta V} = 312$ impulsive maneuvers to enumerate for the trajectory envelope. We then consider the performance over varying values of ΔV_{max} for the initial maneuvers. We use the trajectory segment sampling parameters from the GTOC4 application in Chapter 5. All of these parameters are summarized in Table 4.3. Finally, we assume the set of feasible target objects for new trajectory segments is the set of all asteroids in the GTOC4 problem.

	Search space pruning parameters	
K_α, K_β	Number of discretizations of the spherical angles α and β for possible impulsive maneuver directions	24, 13
$K_{\Delta V}$	Total number of discretizations for impulsive maneuvers (equal to $K_\alpha K_\beta$)	312
N_{BB}	Number of bounding boxes	10
K_{BB}	Number of time intervals for each bounding box	10
	Sampling parameters	
$[\tau_{min}, \tau_{max}]$	Range of allowed segment intercept maneuver times (fraction of segment duration)	$[0.0, 0.0]$
K_τ	Discretizations of τ	1
$[\Delta t_{min}, \Delta t_{max}]$	Range of allowed segment durations	$[0.05, 0.55]$ years
$K_{\Delta t}$	Discretizations of Δt	50

Table 4.3: Search space pruning parameters and trajectory segment sampling parameters for performance analysis of the space pruning procedure.

For the GTOC4 problem there are $N_O = 1436$ asteroids, such that

$$\|\mathcal{F}(\mathbf{x})\| = N_O = 1436 \tag{4.18}$$

The required number of computations for the brute force approach are given in Equations (4.7) and (4.8) and are of constant complexity in the trajectory segment sampling parameters. We can compute their values directly as

$$k_{BF} = K_\tau K_{\Delta t} + \|\mathcal{F}(\mathbf{x})\| \, K_{\Delta t} = 71850 \tag{4.19}$$

$$l_{BF} = \|\mathcal{F}(\mathbf{x})\| \, K_\tau K_{\Delta t} = 71800 \tag{4.20}$$

These are the computations required for expanding a single solution which occurs once per iteration. The required computations for the space pruning approach are given in Equations (4.16) and (4.17). Once again we have constant complexity in the number of required Kepler propagations; however the required Lambert targeting calls is determined by the number of target object intersections which we quantify with the factor f. Therefore for the space pruning approach we have

$$k_{SP} = K_{\Delta V} \left(N_{BB} K_{BB} + 1\right) + \|\mathcal{F}(\mathbf{x})\| \, K_{\Delta t} + K_\tau K_{\Delta t} = 103362 \tag{4.21}$$

$$l_{SP} = \frac{\|\mathcal{F}(\mathbf{x})\| \, K_\tau K_{\Delta t}}{f} = \frac{71800}{f} \tag{4.22}$$

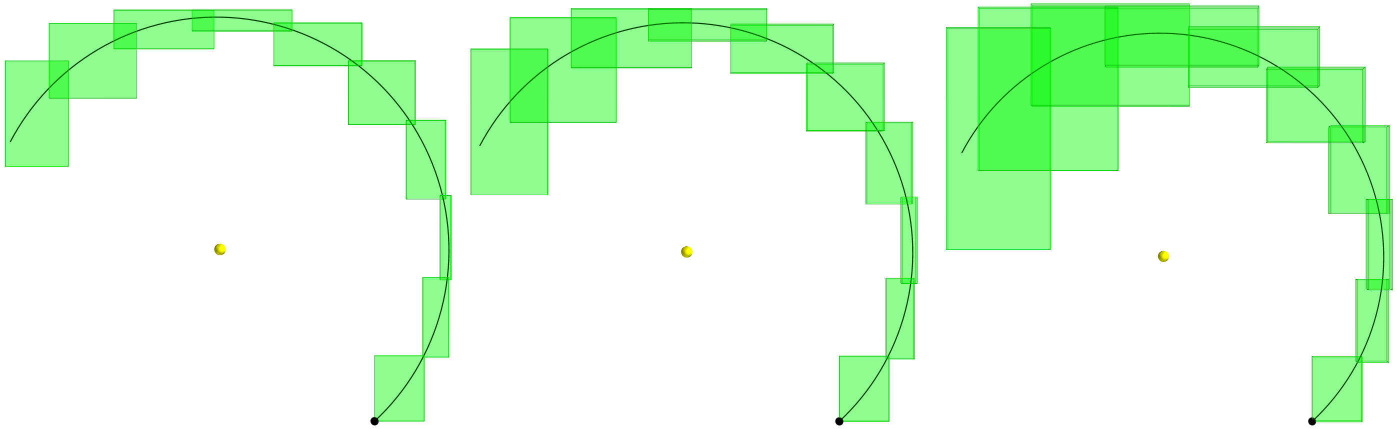
We see that additional Kepler propagations are required for computing the bounding boxes of the trajectory envelope, such that $k_{SP} > k_{BF}$. However, the number of Lambert computations required in the space pruning approach is bounded from above by the brute force approach such that $l_{SP} \leq l_{BF}$. We continue by computing the trajectory envelopes for various values of ΔV_{max} so that we can measure the improvement directly. We assume the spacecraft has initial conditions matching the Earth at $t_{i-1} = 54000$ MJD for the tests. Table 4.4 summarizes the results, and

ΔV_{max}	Intersecting asteroids	Intersecting points	Reduction factor, f
0.25 km/s	11	32	2243.75
0.5 km/s	23	80	897.50
1.0 km/s	46	209	343.54
1.5 km/s	69	421	170.55
2.0 km/s	92	700	102.57
2.5 km/s	125	1048	68.51
3.0 km/s	149	1455	49.35
3.5 km/s	183	1930	37.20
4.0 km/s	224	2479	28.96

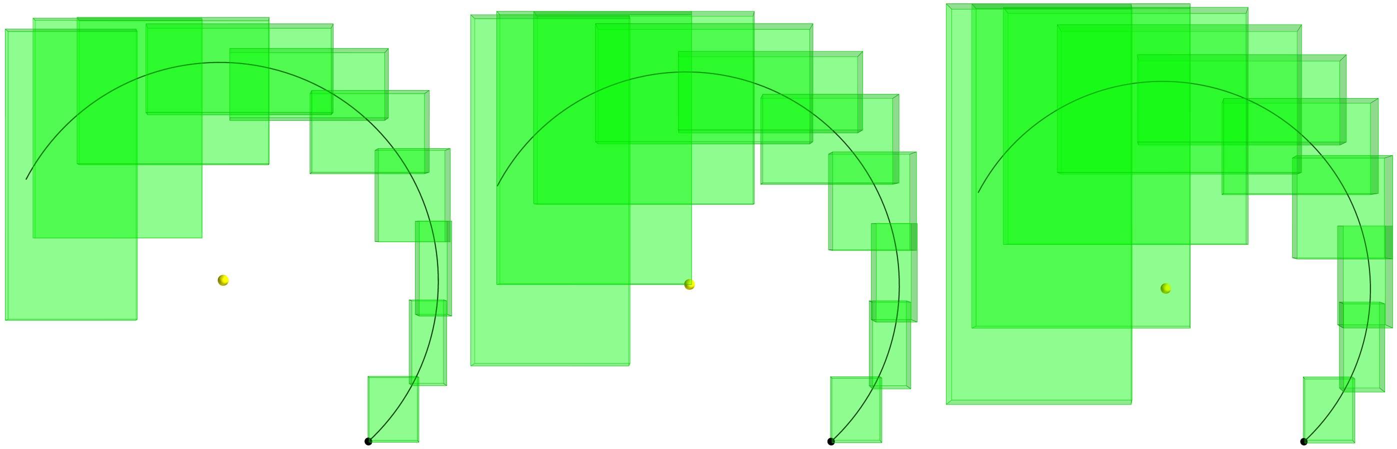
Table 4.4: Results of the search space pruning approach applied to the GTOC4 problem over varying values of ΔV_{max}.

Figure 4.7 shows the trajectory envelope bounding boxes for several of the tests. We see the reduction factor f is largest for small values of ΔV_{max}; this corresponds to the smaller trajectory envelope of the more limited impulse magnitude. As ΔV_{max} increases, we see a lesser but still significant improvement, however. The factor f gives the reduction in overall required Lambert computations, or equivalently the reduction in number of trajectory segments to compute. In this example we see greater than 100× improvements when using the search space pruning approach. We note however that this does not consider the additional Kepler propagations to form the envelope. However, those computations are of fixed complexity while the space pruning improvement scales with problem size. Later analysis considers the improvement in the overall search due to the space pruning procedure.

The critical assumption in the trajectory envelope approach is that the reachable domain of the spacecraft is given by the maximum allowed impulsive maneuver occurring at the beginning of the segment. In our tests here with $\Delta t_{max} = 0.55$ years we have validated this assumption. However, clearly there are cases where this is not true–for example if Δt_{max} corresponds to a full period of the spacecraft, then a



(a) $\Delta V_{max} = 0.25$ km/s (b) $\Delta V_{max} = 0.5$ km/s (c) $\Delta V_{max} = 1.0$ km/s



(d) $\Delta V_{max} = 1.5$ km/s (e) $\Delta V_{max} = 2.0$ km/s (f) $\Delta V_{max} = 2.5$ km/s

Figure 4.7: The bounding boxes associated with trajectory envelopes from the results in Table 4.4.

maneuver at the half-period time will allow the spacecraft to reach regions that a maneuver at the beginning of the segment will not. In such cases the approach can be extended and additional envelopes can be computed for varying maneuver times over the duration of the trajectory segment.

We further note that no effort has been made to tune the search space pruning parameters of Table 4.3 for optimal performance. For example, a larger number of bounding boxes may provide a tighter upper bound on the trajectory envelope. This is an optimization problem unto itself and we leave this analysis to future work. Other bounding shapes (besides boxes) and coordinate systems (such as spherical coordinates) are also possible in the bounding procedure that are worth exploration. Finally, we note that the search space pruning procedure developed here can be combined with other space pruning approaches, such as analytical constraints for specific problem dynamics, for even greater performance improvements.

Chapter 5

Application to Fourth Global Trajectory Optimization Competition

This chapter applies the search methodology developed in Chapter 3 to the fourth annual Global Trajectory Optimization Competition (GTOC) problem. The chapter begins with a description of the GTOC4 problem and its known solutions. The methodology is then applied under varying parameter sets and the results are discussed. Finally, low-thrust finite burn solutions are generated from specific impulsive solutions found in the search.

The GTOC was created in 2005 by the Advanced Concepts Team of the European Space Agency [34]. The competition presents challenging global optimization problems in interplanetary trajectory design, and has enjoyed wide international participation. Each of the seven past competition problems have involved optimizing a mission sequence of some kind [1]. For example, the first GTOC problem required the determination of flyby sequences for gravity assists. The second through fifth and seventh GTOC competitions were each asteroid intercept and/or rendezvous

Problem	**Mission sequence**
GTOC1: "Save the Earth"	multiple gravity assists
GTOC2: "Multiple asteroid rendezvous"	multiple asteroid rendezvous
GTOC3: "Multiple sample return"	multiple asteroid rendezvous, Earth gravity assists allowed
GTOC4: "Asteroids billiard"	multiple asteroid intercept and final asteroid rendezvous
GTOC5: "Penetrators"	multiple asteroid rendezvous and intercept
GTOC6: "Global mapping of Galilean moons"	repeated flybys of four Galilean moons of Jupiter
GTOC7: "Multiple ship mission to main belt asteroids"	multiple spacecraft each with multiple asteroid rendezvous

Table 5.1: Summary of past Global Trajectory Optimization Competition (GTOC) problems and their mission sequences to be optimized [1].

missions where the asteroid sequence was free to be chosen. The sixth GTOC required the design of multiple flybys and gravity assists. The past GTOC problems are summarized in Table 5.1. Each of these mission sequences can be represented by discrete decision variables in the problem formulation. The persistence of these problem types in the competition emphasizes the importance of combinatorial optimization methods for spacecraft trajectory optimization.

The fourth GTOC (GTOC4) occurred in 2009 and was organized by the Interplanetary Mission Analysis team of the Centre National d'Etudes Spatiales de Toulouse, the winners of the third GTOC competition [12]. The mission proposed was entitled "how to maximize the relevance of a rendezvous mission to a given NEA by visiting the largest set of intermediate asteroids". The mission begins when a spacecraft is launched from the Earth. The spacecraft must then flyby a maximum number of asteroids before rendezvousing with a final asteroid. The GTOC4 problem is specifically considered as a benchmark problem for the global search methodology developed in this work.

Maximum launch hyperbolic excess velocity, $v_{\infty_{max}}$	4.0 km/s
Minimum launch date, $t_{0_{min}}$	00:00 January 1, 2015 (57023 MJD)
Maximum launch date, $t_{0_{max}}$	24:00 December 31, 2025 (61041 MJD)
Maximum mission duration, $\Delta t_{mission}$	10 years
m_0	1500 kg
m_{dry}	500 kg
T_{max}	0.135 N
I_{sp}	3000 s

Table 5.2: GTOC4 mission and spacecraft parameters [12].

5.1 Problem Definition

This section summarizes the GTOC4 problem statement from [12]. The spacecraft is launched from Earth with a hyperbolic excess velocity $\mathbf{v}_\infty$ of up to 4.0 km/s in magnitude and in unconstrained direction. The launch must occur within the years of 2015 and 2025, and the total mission duration must not exceed ten years. The spacecraft is equipped with an electric propulsion system, and gravity assists are not allowed during the mission. The mission and spacecraft parameters are summarized in Table 5.2. The Earth and asteroids follow Keplerian orbits about the Sun. The sun's gravitational parameter and Earth's orbital elements are given in Table 5.3. A collection of 1436 near Earth asteroids (NEAs) are given; their orbital elements are provided in Appendix B. Figure 5.1 shows the asteroids' semimajor axis and eccentricity values. The highlighted regions indicate the asteroid types: Atiras, Atens, Apollos and Amors [45].

After being launched from Earth, the spacecraft must flyby a maximum number of intermediate asteroids and then rendezvous with a final asteroid. Each asteroid may be visited at most once (intercept or rendezvous). An intercept requires

Sun gravitational parameter, μ (km^3/s^2)	1.32712440018E11
Semimajor axis, a (AU)	0.999988049532578
Eccentricity e	1.671681163160E-2
Inclination i (deg)	0.8854353079654E-3
Longitude of ascending node Ω (deg)	175.40647696473
Argument of periapsis ω (deg)	287.61577546182
Mean anomaly at epoch M (deg)	257.60683707535
Epoch t (MJD)	54000

Table 5.3: The sun's gravitational parameter and Earth orbital elements for GTOC4 problem in J2000 heliocentric ecliptic reference frame [12].

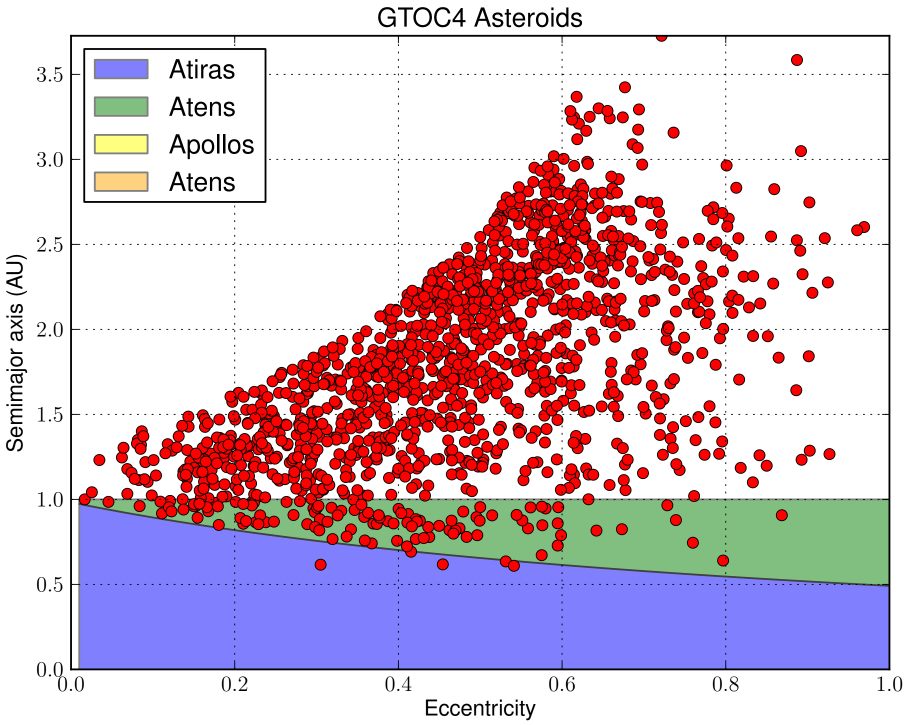


Figure 5.1: The set of 1436 asteroids for the GTOC4 problem. The highlighted regions show combinations of semimajor axis and eccentricity for Atiras, Atens, Apollos and Amors near-Earth asteroids [45].

a match of position between the spacecraft and asteroid, while a rendezvous requires a match of both position and velocity. The objective to be maximized is the number of intermediate asteroids. The provided objective function is given by

$$J = \sum_{j=1}^{n} \alpha_j \tag{5.1}$$

where n is the total number of asteroids provided, and $\alpha_j \in \{0, 1\}$ denotes the number of times asteroid j is visited during the mission, excepting the final rendezvous. That is, the final asteroid j_f corresponds to $\alpha_{j_f} = 0$. Therefore the objective function J cannot exceed $n - 1$. If multiple solutions have the same objective value J, the best solution is that which maximizes the final mass, given by the secondary objective function

$$K = m_f \tag{5.2}$$

5.1.1 GTOC4 Augmented Impulsive Tour Model

We use the augmented impulsive tour model developed in Section 2.2.6 for the GTOC4 problem. Following the notation defined there and defining the objective for minimization rather than maximization, we define the objective as

$$J(\mathbf{x}) = -(n_s - 1) \tag{5.3}$$

which corresponds to the number of intermediate asteroids. The mission launch timing and duration constraints are given by

$$t_{0_{min}} \leq t_0 \leq t_{0_{max}} \tag{5.4}$$

$$t_{n_s} \leq t_0 + \Delta t_{mission} \tag{5.5}$$

The spacecraft begins the mission at Earth with a constrained hyperbolic excess velocity. Its initial conditions are constrained as

$$\mathbf{r}(t_0) = \mathbf{r}_{earth}(t_0) \tag{5.6}$$

$$\|\mathbf{v}(t_0) - \mathbf{v}_{earth}(t_0)\| \leq v_{\infty_{max}} \tag{5.7}$$

Each asteroid may only be visited once in the tour. Rather than write explicit constraints on the sequence decision variables $s_1 \ldots s_{n_s}$, we instead specify the set of feasible target objects for solution construction according to Section 3.4 as

$$\mathcal{F}(\mathbf{x}) = O \setminus \{s_1 \ldots s_{n_s}\} \tag{5.8}$$

where O is the set of all GTOC4 asteroids. The feasible target objects $\mathcal{F}(\mathbf{x})$ are evaluated every time new segments are added to an existing tour. For a given trajectory segment i of the tour then, this constrains the feasible target objects that may be chosen as

$$s_i \in O \setminus \{s_1 \ldots s_{i-1}\} \tag{5.9}$$

The GTOC4 problem requires intercepting a number of intermediate asteroids before rendezvousing with a final asteroid. When we add new trajectory segments onto an existing tour then, we can attempt to either intercept or rendezvous with the next target. If we intercept the next target, then we say the result is a partial solution, since it may still be expanded to produce a longer tour. If however we rendezvous with the next target, the solution is complete since it satisfies the original problem statement and can no longer be expanded. During solution construction, we create both intercept and rendezvous segments to the next target object s_i, and the feasible segments are kept.

We restate the augmented impulsive tour model developed in Section 2.2.6 here with the additional constraints developed for GTOC4. The GTOC4 augmented impulsive tour model is thus

$$\begin{aligned}
\text{Determine} \quad & \mathbf{x} = [t_0, \mathbf{z}_0,\ s_1 \ldots s_{n_s},\ \mathbf{y}_1 \ldots \mathbf{y}_{n_s}] && (5.10)\\
\text{minimizing} \quad & J(\mathbf{x}) = -(n_s - 1) && (5.11)\\
\text{subject to} \quad & \mathbf{z}(t) = \mathbf{f}(t, \mathbf{x}) \implies \Delta\mathbf{V}_1 \ldots \Delta\mathbf{V}_{N_{\Delta V}} && (5.12)\\
& m(t_{n_s}) \geq m_{dry} && (5.13)\\
& (t_{fb_0})_0 \geq t_0 && (5.14)\\
& (t_{fb_f})_{N_{\Delta V}} \leq t_{n_s} && (5.15)\\
& (t_{fb_f})_i \leq (t_{fb_0})_{i+1} \qquad \text{for } i \in 1 \ldots N_{\Delta V} - 1 && (5.16)\\
& t_{0_{min}} \leq t_0 \leq t_{0_{max}} && (5.17)\\
& t_{n_s} \leq t_0 + \Delta t_{mission} && (5.18)\\
& \mathbf{r}(t_0) = \mathbf{r}_{earth}(t_0) && (5.19)\\
& \|\mathbf{v}(t_0) - \mathbf{v}_{earth}(t_0)\| \leq v_{\infty_{max}} && (5.20)\\
& \mathbf{r}(t_i) = \mathbf{r}_{s_i}(t_i) \qquad \text{for } i = 1 \ldots n_s && (5.21)\\
& \mathbf{v}(t_{n_s}) = \mathbf{v}_{s_{n_s}}(t_{n_s}) && (5.22)
\end{aligned}$$

The dynamics $\mathbf{f}(t, \mathbf{x})$ are computed according to the augmented impulsive tour model, where ballistic arcs are propagated according to Kepler's problem and impulsive maneuvers are found as the solution to Lambert's problem. Equations (5.21) define the intermediate intercepts of the tour, and for $i = n_s$ Equations (5.21) and (5.22) define the final rendezvous for complete solutions. These constraints determine the required maneuvers (whether a rendezvous maneuver is necessary) and are implicitly satisfied in the computation of the dynamics. Finally, Equations (5.14) through (5.16) constrain estimates of representative finite burn maneuvers from overlapping in time or exceeding the mission time bounds. These estimates are based on the development in Section 2.2.5 and include the effects of gravity losses.

5.2 Best Known Solutions

We choose to apply the methodology to the GTOC4 problem in part because there exist a collection of known solutions to the problem generated using a variety of methods. We use these existing solutions as a benchmark to judge the performance of our own search algorithm. The final results of the GTOC4 competition are shown in Table 5.4. The winning solution was found by Moscow State University and visited 44 intermediate intercepts before finally rendezvousing with asteroid 2000SZ162.

We emphasize that these solutions are low-thrust finite burn trajectories, whereas the solutions we generate during the search use impulsive maneuvers. Although we constrain the impulsive solutions based on estimates of representative finite burns in an effort to ensure a finite burn conversion is feasible, a direct comparison is still not appropriate. We however reference the objectives achieved by these solutions in the discussion of our results. In later sections we find optimal low-thrust finite burn trajectories based on these impulsive solutions that are feasible to the original GTOC4 problem statement.

5.3 Results

In this section we apply the tabu search algorithm developed in Chapter 3 to the GTOC4 problem. We first examine results of a base case for a given set of search parameters. We then conduct a sensitivity analysis to determine the effects of varying these parameters and components of the search algorithm. For each case we generate a collection of results corresponding to a range of launch times t_0. Unless indicated otherwise, we generate results for 1,024 launch epochs evenly spaced over the allowed 10 year launch window and run each search for 2 hours. Each case therefore requires 2,048 hours of compute time. This is summarized in Table 5.5.

Rank	Team name	J	Final mass (kg)	Duration (years)	Rendezvous asteroid
1	Moscow State University	-44	553.46	10	2000SZ162
2	The Aerospace Corporation	-44	516.83	10	2000SZ162
3	Advanced Concepts Team, ESA	-42	511.45	10	2008UA202
4	DEIMOS Space	-39	605.44	10	2006BZ147
5	GMV	-39	516.30	10	2007YF
6	Jet Propulsion Laboratory	-38	515.87	10	138911
7	Politecnico di Torino, Universita di Roma La Sapienza	-36	574.44	10	2006QQ56
8	University of Texas at Austin, Odyssey Space Research, ERC Incorporated	-32	639.86	9.69	2006UB17
9	University of Glasgow, University of Strathclyde	-29	715.21	9.98	2006QQ56
10	Thales Alenia Space	-27	533.25	10	2006QQ56
11	University of Trento	-26	721.73	9.73	2006UB17
12	University of Bremen, Politecnico di Milano	-26	577.97	9.82	2008GM2
13	Moscow Aviation Institute, Research Institute of Applied Mechanics and Electrodynamics	-24	720.62	10	2007YF
14	Georgia Institute of Technology	-24	500.27	9.5	2008UA202
15	TOMLAB	-22	615.22	9.65	2006XP4
16	VEGA	-20	653.07	10	2008UA202
17	DLR German Space Operations Center, Aachen University of Applied Sciences	-20	635.09	10	2005BG28
18	Team Astroshape	-20	524.48	10	2006SV5
19	DLR Institute of Space Systems	-19	592.35	10	138911
20	Tsinghua University	-18	539.98	10	138911
21	University of Missouri	-15	836.06	10	2005CD69
22	Beijing University of Aeronautics and Astronautics	-13	651.87	9.98	2006RJ1
23	Texas A&M University	-12	697.93	10	2006UB17

Table 5.4: Final results of the fourth Global Trajectory Optimization Competition (GTOC4) [11].

Minimum launch epoch	Maximum launch epoch	Number of runs	Run time
57023 MJD 00:00 January 1, 2015	61041 MJD 24:00 December 31, 2025	1024	2 hours

Table 5.5: For each case, a collection of 1024 runs are generated over a range of launch epochs. Each run executes for 2 hours, requiring 2048 compute hours in total.

All results were computed on the Stampede supercomputer at the Texas Advanced Computing Center (TACC) at the University of Texas at Austin [2]. The author thanks TACC for supporting this work.

5.3.1 Base Case

We define a set of parameters for the tabu search algorithm in Table 5.6. We generate new trajectory segments in the search according to the sampling parameters provided. For this case, intercept maneuvers always occur at the beginning of the trajectory segment, and we consider segment durations from 0.05 to 0.55 years with 50 discretizations. When a new best solution is found, we decrease the goal objective (which in part defines the guiding objective) by $\Delta J_{goal} = 1$. The guiding objective penalizes excess ΔV over the defined budget with a weight of 1. A tabu tenure of 10 iterations is used to prevent cycling among similar solutions. The parameters for the dynamic neighborhood selection procedure are also shown.

We first examine the runs in terms of their best solutions found. Recall that a partial solution consists only of intermediate intercepts, while a complete solution also includes a rendezvous with a final target asteroid. Figure 5.2 and Table 5.7 shows the results for the 1024 runs of the base case. We see that we achieve better objectives for partial solutions than for complete solutions. This is expected as the rendezvous conditions are more difficult to satisfy; the target asteroid must be in a similar orbit to the spacecraft in order for a final rendezvous maneuver to be feasible. We further see approximately normal distributions of the best objectives achieved, with no particular range of launch epochs more or less favorable. For complete solutions, the median best objective found is $J = -42$, with the best solutions found to be $J = -45$. The $J = -45$ solutions exceed the best known GTOC4 solutions–we note again however that these are impulsive solutions rather

	Sampling parameters	
$[\tau_{min}, \tau_{max}]$	Range of allowed segment intercept maneuver times (fraction of segment duration)	$[0.0, 0.0]$
K_τ	Discretizations of τ	1
$[\Delta t_{min}, \Delta t_{max}]$	Range of allowed segment durations	$[0.05, 0.55]$ years
$K_{\Delta t}$	Discretizations of Δt	50
	Guiding objective parameters	
ΔJ_{goal}	Parameter for updating $\frac{dJ}{dt}^{\star}$ when a new best solution is found	1
w_m	Budget penalty weight for mass in guiding objective	0.0
$w_{\Delta V}$	Budget penalty weight for ΔV in guiding objective	1.0
	Tabu search parameters	
$N_{\mathcal{T}}$	Tabu tenure	10
$[h_{min}, h_{max}]$	Allowed range of neighborhood parameter h for intensification or diversification	$[3, 7]$
Δh_{escape}	Change in neighborhood parameter h for escape procedure	10
$N_{cooldown}$	Cool down period: number of iterations between allowed changes to the neighborhood	25
N_{stall}	Stall period: number of iterations over which the stall condition is evaluated	25
N_{escape}	Escape period: number of iterations over which the escape condition is evaluated	50
$\underline{R}$	Stall threshold: a stall occurs when the range of solution lengths is below $\underline{R}$ for N_{stall} iterations	5
$\bar{R}$	Relaxation threshold: when the range of solution lengths is above $\bar{R}$, the neighborhood is relaxed	15

Table 5.6: Tabu search algorithm parameters for the base case.

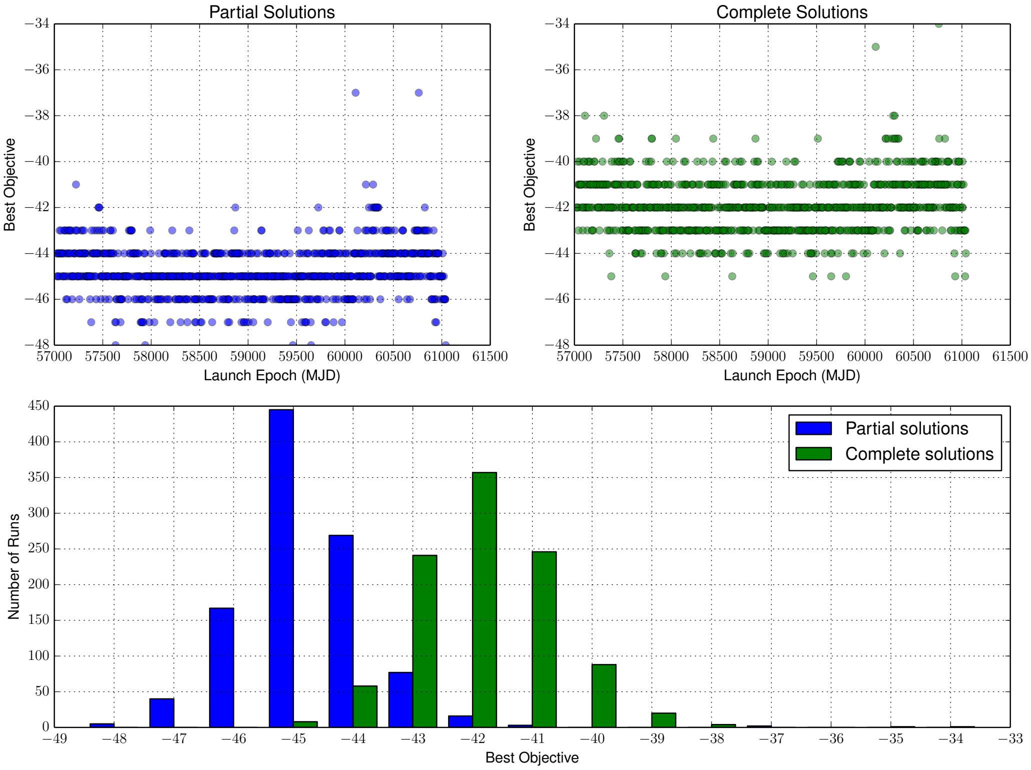


Figure 5.2: Base case: Best partial and complete solutions found for 1024 runs over launch dates from 2015 to 2025.

	Minimum	**Median**	**Maximum**
Base case (partial)	-48	-45	-37
Base case (complete)	-45	-42	-34

Table 5.7: Base case: the minimum, median and maximum best solutions found over the 1024 runs.

Rendezvous asteroid	**Best objective, J** GTOC4 results	Base case (1024 runs)	Base case: run #409/1024
2000SZ162	-44	-44	-42
2008UA202	-42	-45	-42
2006BZ147	-39	-45	-45
2007YF	-39	-45	-42
138911	-38	-44	-38
2006QQ56	-36	-45	-42
2006UB17	-32	-44	-41
2008GM2	-26	-44	-42
2006XP4	-22	-43	–
2005BG28	-20	-44	–
2006SV5	-20	-42	-42
2005CD69	-15	-44	-39
2006RJ1	-13	-44	–

Table 5.8: Comparison of best solutions found in the GTOC4 competition results and the base case [11].

than low-thrust finite burn solutions.

We examine the entire population of complete solutions generated in the base case and find solutions corresponding to each of the final rendezvous asteroids from the GTOC4 competition results. These are shown in Table 5.8. We see that in all cases we find results that meet or exceed the GTOC4 competition results. In fact, for only two of the rendezvous asteroids do the results not meet the GTOC4 winning objective of $J = -44$. This demonstrates the broad set of good solutions the search is able to find.

We now investigate an individual run of the 1024 runs executed for the base case. We choose run #409/1024 which corresponds to a launch epoch of $t_0 = 58629.41$ MJD. This is one of the several runs that found complete solutions for $J = -45$. We find that this run alone finds solutions rendezvousing with 10 of the 13 final asteroids from the GTOC4 results, also shown in Table 5.8. Fig-

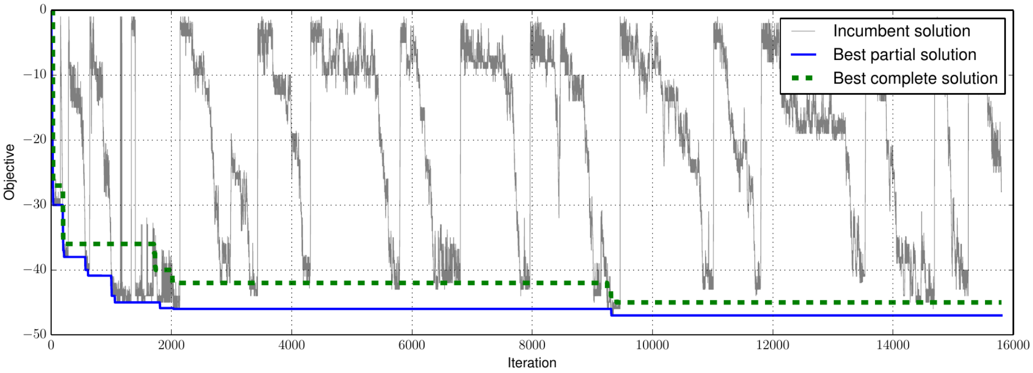


Figure 5.3: Base case: run #409/1024. The objective history for the incumbent solution and best found partial and complete solutions are shown.

ure 5.3 shows the objectives achieved over the iteration history. The search executes more than 15,000 iterations over the two hour run time. The search quickly finds complete solutions better than $J = -40$ and finds $J = -45$ complete solutions approximately half way through. We see an oscillatory behavior in the incumbent solution objective. Recall that the objective corresponds to the solution length n_s, or equivalently the depth in the search tree. These oscillations therefore represent a diverse exploration of the search tree, as solutions of varying lengths are visited and stalls are quickly broken. We attribute this diversity of the search in part to the dynamic neighborhood selection procedure that modifies the neighborhood when stalls are detected. The impact of dynamic neighborhood selection is investigated further in Section 5.3.4.

We now examine one of the $J = -45$ solutions generated by this run. Figure 5.4 shows the trajectory of a solution rendezvousing with the asteroid 2006BZ147. The full tour itinerary is shown in Table 5.9. The mass history over the trajectory closely follows the linear mass budget. We see portions of the trajectory where the spacecraft uses more mass than budgeted; the partial solutions ending at these segments would be penalized during the search. However, they are still feasible and

no.	t (MJD)	mass (kg)	Asteroid	no.	t (MJD)	mass (kg)	Asteroid
0	58629.41	1500.00	Earth	1	58713.42	1500.00	2006QV89
2	58815.69	1470.01	2003YT70	3	58922.79	1449.25	2008CL20
4	59003.68	1405.84	2005ED318	5	59062.81	1392.57	2005NW44
6	59162.22	1367.57	2007TK15	7	59199.40	1351.71	2005GA120
8	59265.91	1343.26	2006KQ1	9	59406.16	1300.46	2001GO2
10	59523.95	1273.81	2000RN77	11	59604.68	1237.90	2003SW130
12	59674.94	1221.59	162173	13	59792.74	1183.36	2007DS84
14	59891.76	1139.15	2005BC	15	59969.68	1116.96	2008SW7
16	60025.00	1094.31	2003WY153	17	60113.15	1075.84	1998DK36
18	60186.68	1058.79	2002JR100	19	60256.89	1032.36	2008NQ3
20	60326.59	1007.38	2006RJ7	21	60396.51	988.81	2007US12
22	60481.29	961.61	2004YC	23	60532.90	947.45	2005CN
24	60635.94	922.78	2007LF	25	60790.26	910.34	2004TP1
26	60853.27	902.49	2005XY4	27	60912.49	881.65	162157
28	60986.19	860.20	2005GY8	29	61071.24	831.07	68372
30	61162.56	806.94	2008TC3	31	61236.31	794.12	2004JN1
32	61298.81	771.87	175729	33	61368.96	750.65	2007YF
34	61449.31	739.09	2006UB17	35	61512.08	718.63	2005NE21
36	61557.33	698.63	2004RU109	37	61634.61	685.94	2000UG11
38	61734.14	664.80	2004PR92	39	61800.31	645.00	162416
40	61874.01	630.85	2006WQ29	41	61933.24	607.81	2007RE2
42	61959.06	601.32	2001SE270	43	62032.77	588.42	2002XV90
44	62103.03	574.49	2007DS7	45	62202.56	545.42	2004CZ1
46	62273.00	506.24	2006BZ147				

Table 5.9: Base case: run #409/1024. The tour itinerary is shown for a $J = -45$ solution rendezvousing with asteroid 2006BZ147.

ultimately the search expands the trajectory to the complete solution shown.

Overall we find that the search produces a diverse population of good solutions to the GTOC4 problem using the base case parameters. We find trajectories exceeding the best known solutions of the GTOC4 problem. We now continue by varying aspects of the search and studying the effects in comparison to this base case.

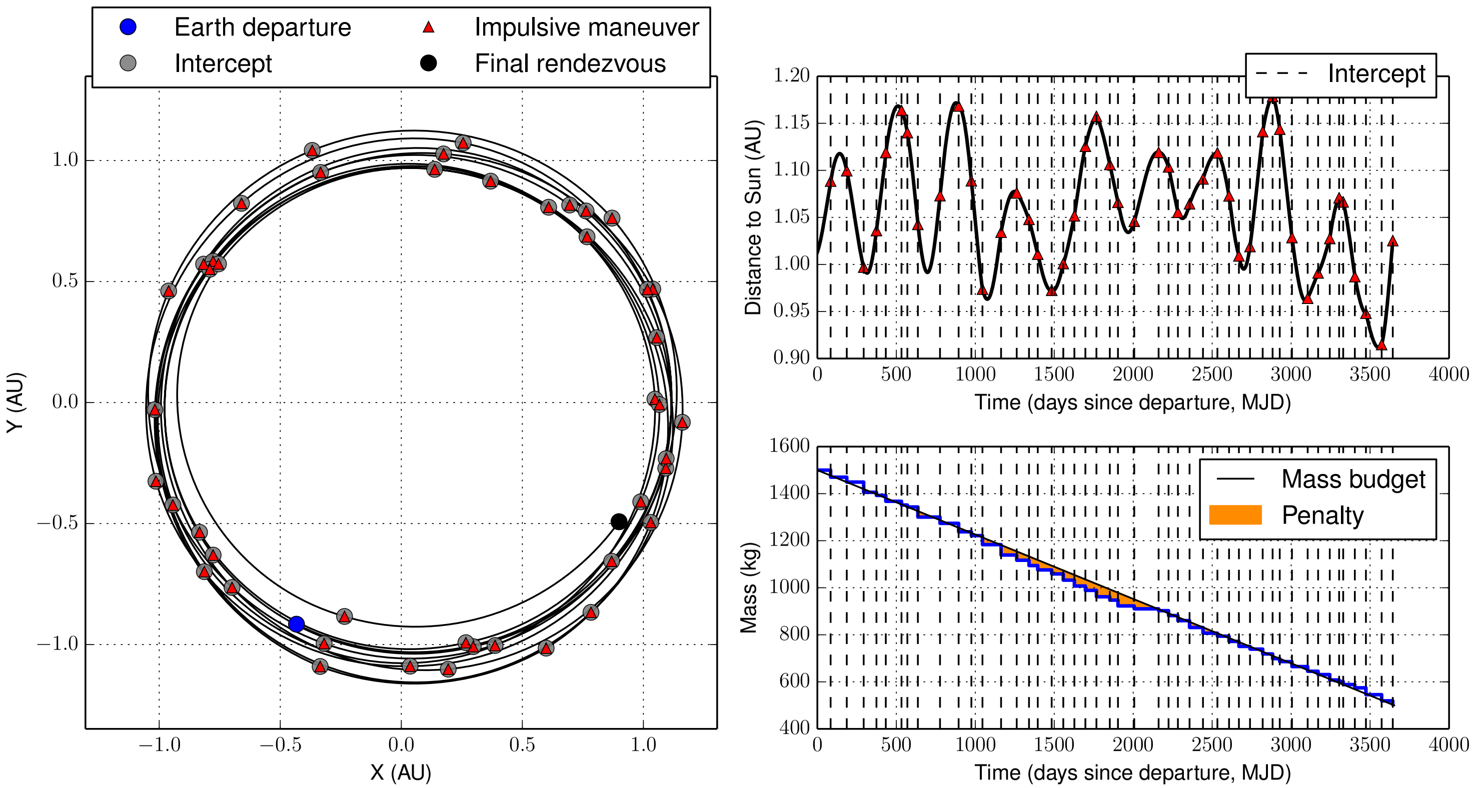
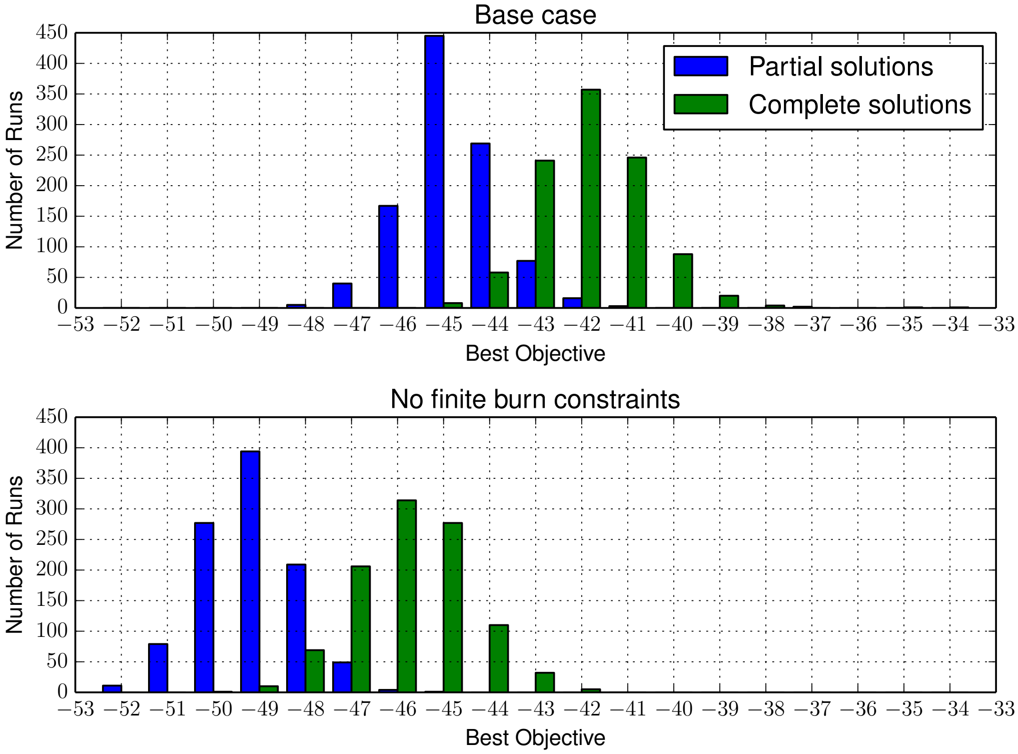


Figure 5.4: Base case: run #409/1024. The trajectory is shown for a $J = -45$ solution rendezvousing with asteroid 2006BZ147.

5.3.2 Finite Burn Constraints

The finite burn constraints in the augmented impulsive tour model (Equations (5.14) through (5.16)) ensure that estimates for finite burn maneuvers representing each impulsive maneuver do not overlap in time or occur outside the mission time of flight. These finite burn estimates are developed in Section 2.2.5 and include the effects of gravity losses. The goal of these constraints is to restrict the search to find only impulsive solutions that can likely be feasibly converted to low-thrust finite burn trajectories.

The additional constraints further limit the feasible solution space of the problem. We run an additional case without these constraints in order to study their effects. We use the base case search parameters of Table 5.6, but ignore the finite burn constraints in the augmented impulsive tour model. Figure 5.5 shows the best solutions found over 1024 runs in comparison to the base case. The differences are significant: the median best complete solution ignoring finite burn constraints is $J = -46$, better than the best overall complete solution found in the base case. Further, we find complete solutions visiting up to 50 intermediate asteroids ($J = -50$). These solutions, although not practical, are impressive compared to the known GTOC4 solutions. We can also examine the quantity and diversity of solutions found. Recall a family of trajectories is identified by its asteroid sequence $s_i \dots s_{n_s}$. Figure 5.6 compares the number of trajectory families found for complete solutions to the base case. We see at least an order of magnitude more solutions over each range of objectives, with some ranges seeing a nearly 5 order of magnitude difference. The disparity is greatest for objectives near zero–these solutions correspond to short sequences. The spacecraft is less efficient when its mass is maximum early in a trajectory; for the early maneuvers then the finite burn estimates will be of longer duration, and the finite burn constraints will eliminate more solutions. This explains



Base case

No finite burn constraints

Figure 5.5: The best solutions found for the base case and the case ignoring finite burn constraints.

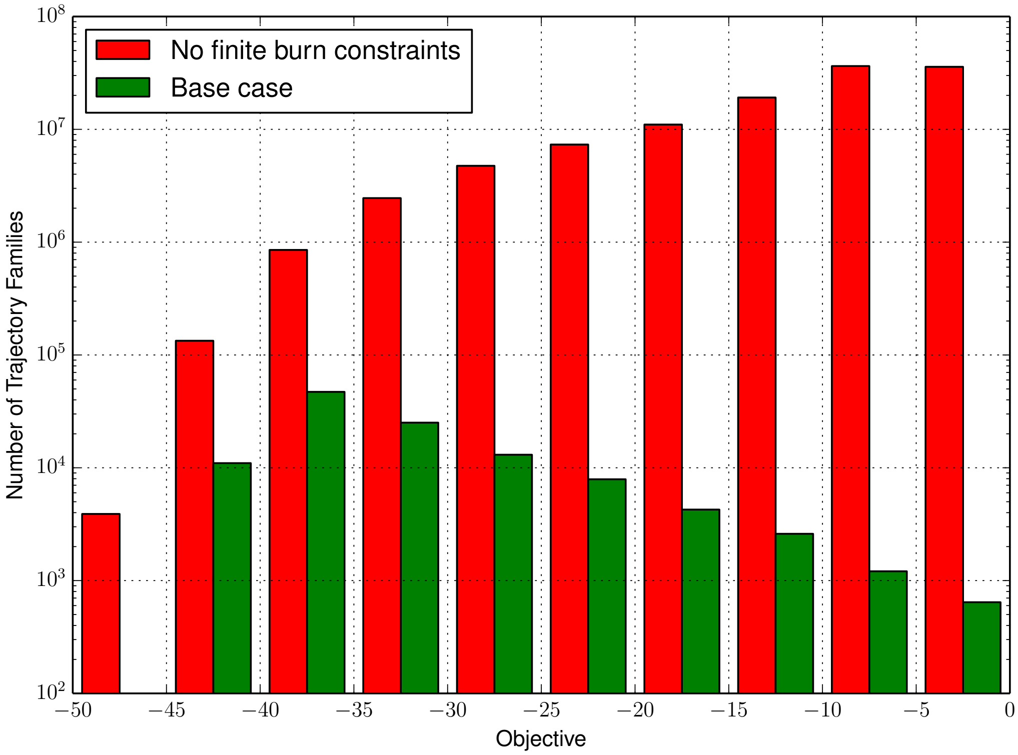


Figure 5.6: The number of trajectory families found for complete solutions in the base case and the case ignoring finite burn constraints.

in part these larger differences.

The finite burn constraints prune at least an order of magnitude of the solutions away over every range of objectives. These pruned solutions represent better objective values; however they are not likely to have feasible low-thrust finite burn counterparts and are thus not of interest in the search. Further, the search tree is smaller in size when such solutions are pruned, which increases the efficiency of the search.

5.3.3 Search Space Pruning

The search space pruning procedure developed in Chapter 4 is designed to accelerate the generation of new solutions by discarding infeasible trajectory segments before they are computed. This in turn accelerates the computation of each iteration and allows the search to explore a larger region of the search space in a fixed time. We now disable the space pruning procedure and re-run results to measure the performance improvement in comparison to the base case. We otherwise use the same base case search parameters of Table 5.6.

We first run results for the same fixed run time of 2 hours and consider the performance in terms of the number of iterations computed during that time. Figure 5.7 shows a comparison against the base case with space pruning enabled. We see that in all runs more iterations are computed with space pruning enabled than not. A median of 10.5× more iterations are computed when space pruning is enabled. This order of magnitude improvement allows the search to explore and find more improved solutions for the same fixed compute time. Over 95% of the runs see at least a 5× improvement. The improvement directly impacts the quality of solutions found. Figure 5.8 shows a comparison of the best solutions found in comparison to the base case. The differences are significant; without space pruning, the median objective found for complete solutions is only $J = -38$, much worse

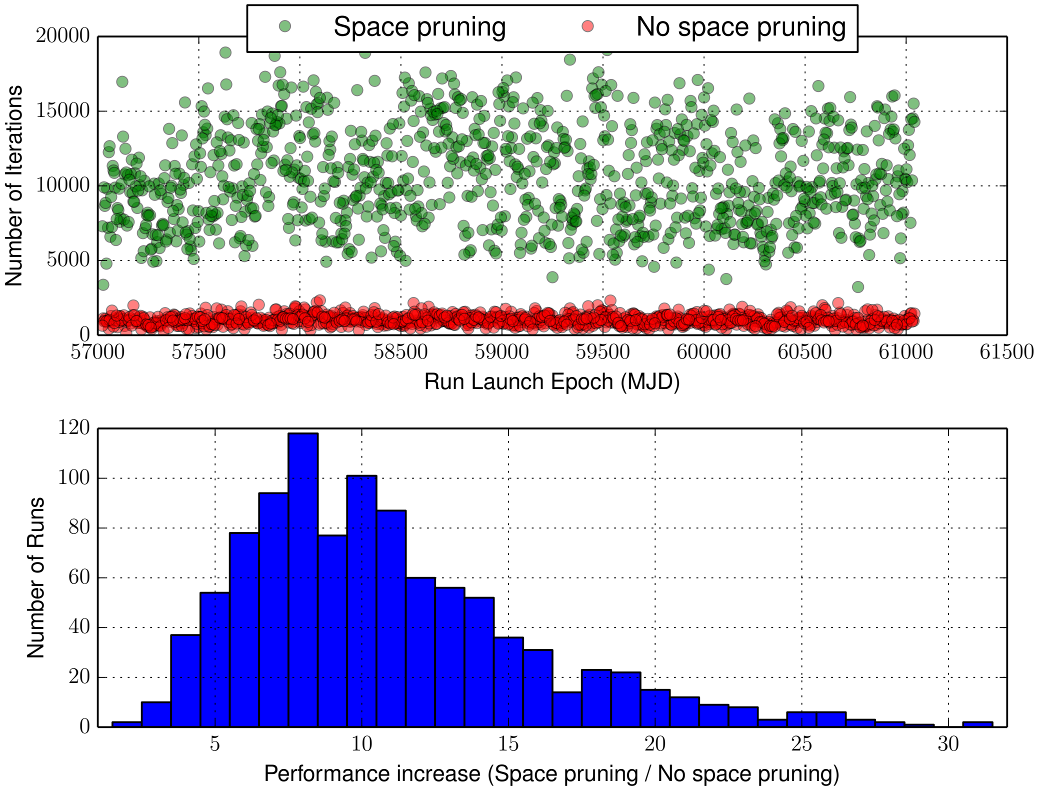
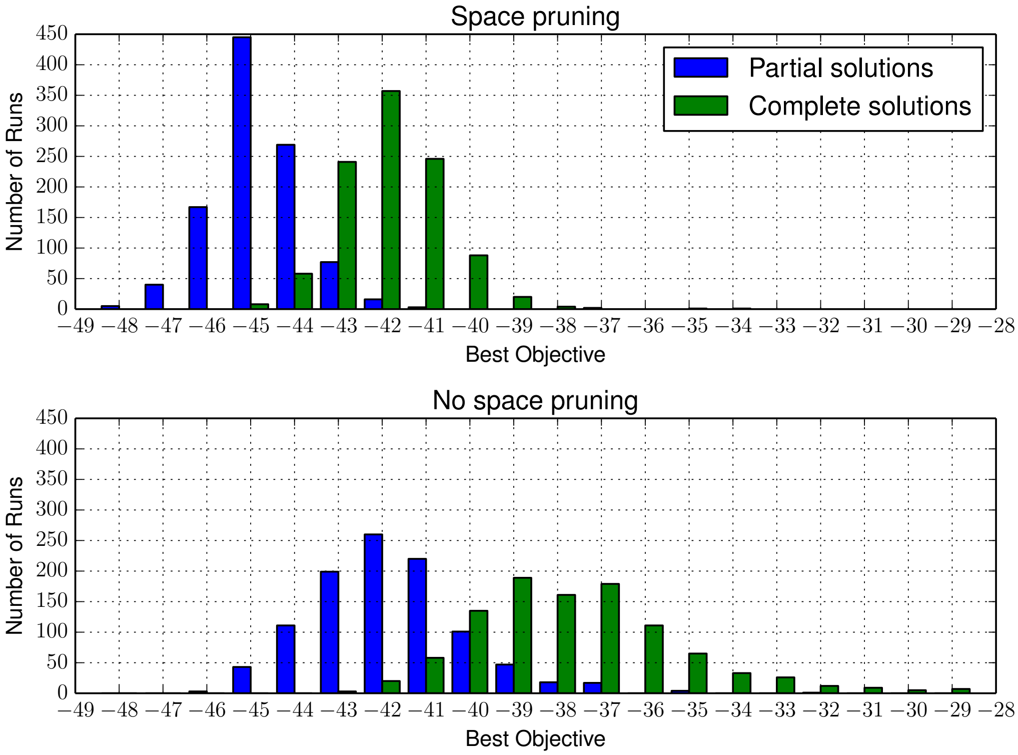


Figure 5.7: Performance of base case (search space pruning) versus case with space pruning disabled for 1024 runs. The run time is constrained to 2 hours. A median of 10.5× more iterations are computed with space pruning enabled, with 95% of runs seeing at least a 5× improvement.


Space pruning

No space pruning

Figure 5.8: Results of base case (search space pruning) versus case with space pruning disabled for 1024 runs. The run time is constrained to 2 hours. The best partial and complete solutions are shown.

than the base case value of $J = -42$.

We run another comparison against the base case, but this time fix the runs at 1,000 iterations total with an unrestricted run time. We then compare the performance in terms of the required run time. Figure 5.9 shows the results. All of the runs with space pruning enabled complete in a shorter run time than those without space pruning. The median runtime speedup of the space pruning approach is 11.0×. Again, we see more than an order of magnitude improvement. Over 99% of the runs see a greater than 5× runtime speedup.

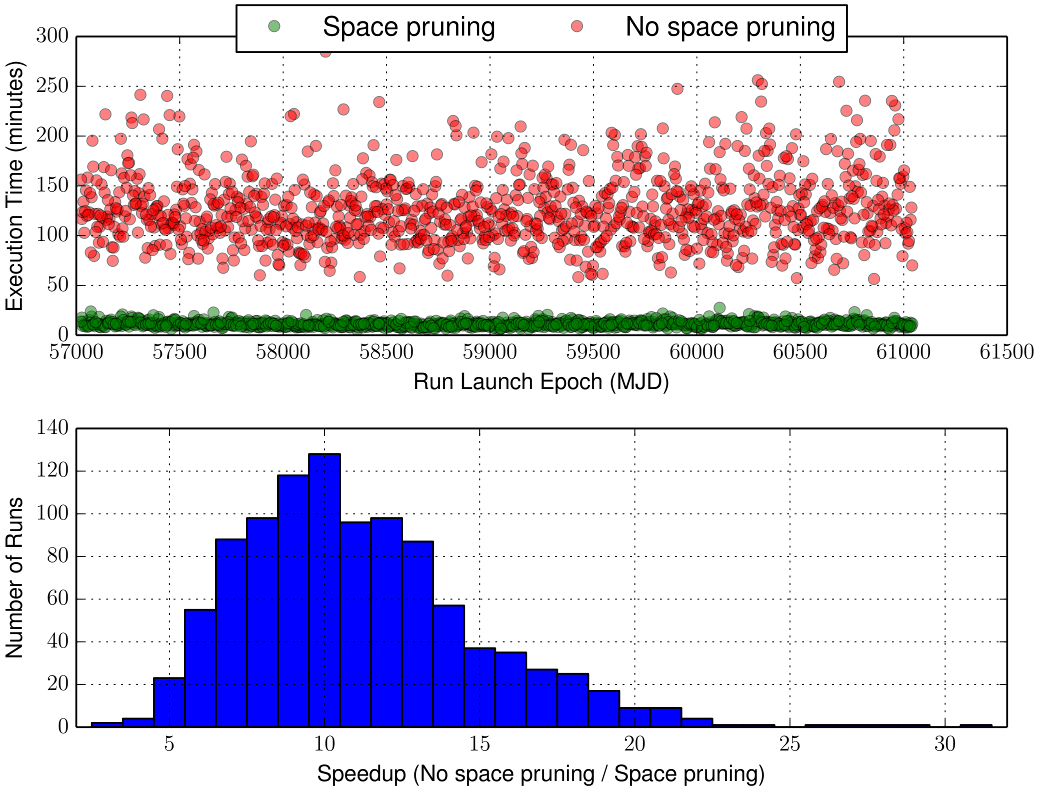


Figure 5.9: Performance of base case (search space pruning) versus case with space pruning disabled for 1024 runs. 1000 iterations are computed and the run time is unconstrained. The median runtime speedup is 11.0× with 99% of the runs seeing at least a 5× speedup.

The search space pruning procedure provides a median order of magnitude performance improvement to the search, with practically all runs seeing at least a 5× improvement. We can either compute an order of magnitude more iterations in the same allowed run time, or compute the same number of iterations an order of magnitude faster. This provides a significant boost to the search performance, especially important in the context of limited compute resources or a limited time during which to solve a problem. We see a distribution of performance gains rather than a fixed improvement due to the variability in the number of infeasible trajectory segments that are pruned during solution construction. We expect to see a larger improvement when fewer feasible solutions exist, since more solutions can be discarded prior to computation. When most of the candidate solutions are feasible, the space pruning approach is of less benefit.

5.3.4 Dynamic Neighborhood Selection

The dynamic neighborhood selection procedure developed in Section 3.5.2 dynamically intensifies and diversifies the search in an effort to provide improved overall solutions compared to statically defined neighborhoods. It achieves this by maintaining a diverse exploration of the search tree, breaking any stalls about particular regions of the search space that may occur. In this section we compare the dynamic neighborhood selection approach to various static neighborhoods.

Recall that we use the restricted best-first neighborhood defined in Section 3.2.2 in the tabu search algorithm. This neighborhood has one parameter h that dictates the neighborhood's size, in terms of how far upward in the tree to move before finding neighboring leaf nodes to expand. The dynamic neighborhood selection procedure adjusts h throughout the search. A static neighborhood instead maintains a fixed value of h.

We now run cases for various fixed values of h. These cases use the same search parameters as the base case (Table 5.6) but do not dynamically update the neighborhood. The results for several trial values of h are shown in Figure 5.10. We see that the $h = 3$ case significantly underperforms the base case, with a median best complete solution of only $J = -31$. The performance improves as h is increased. The $h = 5$ is a small improvement with a median best complete solution of $J = -32$. The improvement continues with the $h = 20$ case giving a median best complete solution of $J = -41$, nearly as good as the base case with dynamic neighborhood selection enabled ($J = -42$). From these metrics then, it appears that the results steadily improve as h is increased.

We now compare the extreme case of $h = \infty$ with the base case directly. Figure 5.11 shows the results. The distributions of best solutions found for each run appear nearly identical, with the $h = \infty$ case having several runs achieving best complete solutions of only $J = -28$ while all runs of the base case (dynamic neighborhood selection) achieve at least $J = -34$. However, the results are further distinguished when we consider the quantity and variety of solutions found rather than just the best single solution of each run. Recall that we define a family of trajectories by their target object sequence $s_1 \ldots s_{n_s}$. Figure 5.12 compares the $h = \infty$ case against the dynamic neighborhood selection case based on the number of trajectory families found for complete solutions. We see many more families of solutions are found for the dynamic neighborhood selection case. This is an important result, as it indicates the search is providing a more varied set of trajectory options in the population of final solutions. We attribute this to the strategic intensification and diversification of the search provided by the dynamic neighborhood selection procedure. We can quantify this diversity in part by the variation in the incumbent solution lengths over the iterations. Figure 5.13 shows that the standard deviation of incumbent solution lengths is much higher for the dynamic neighborhood selection

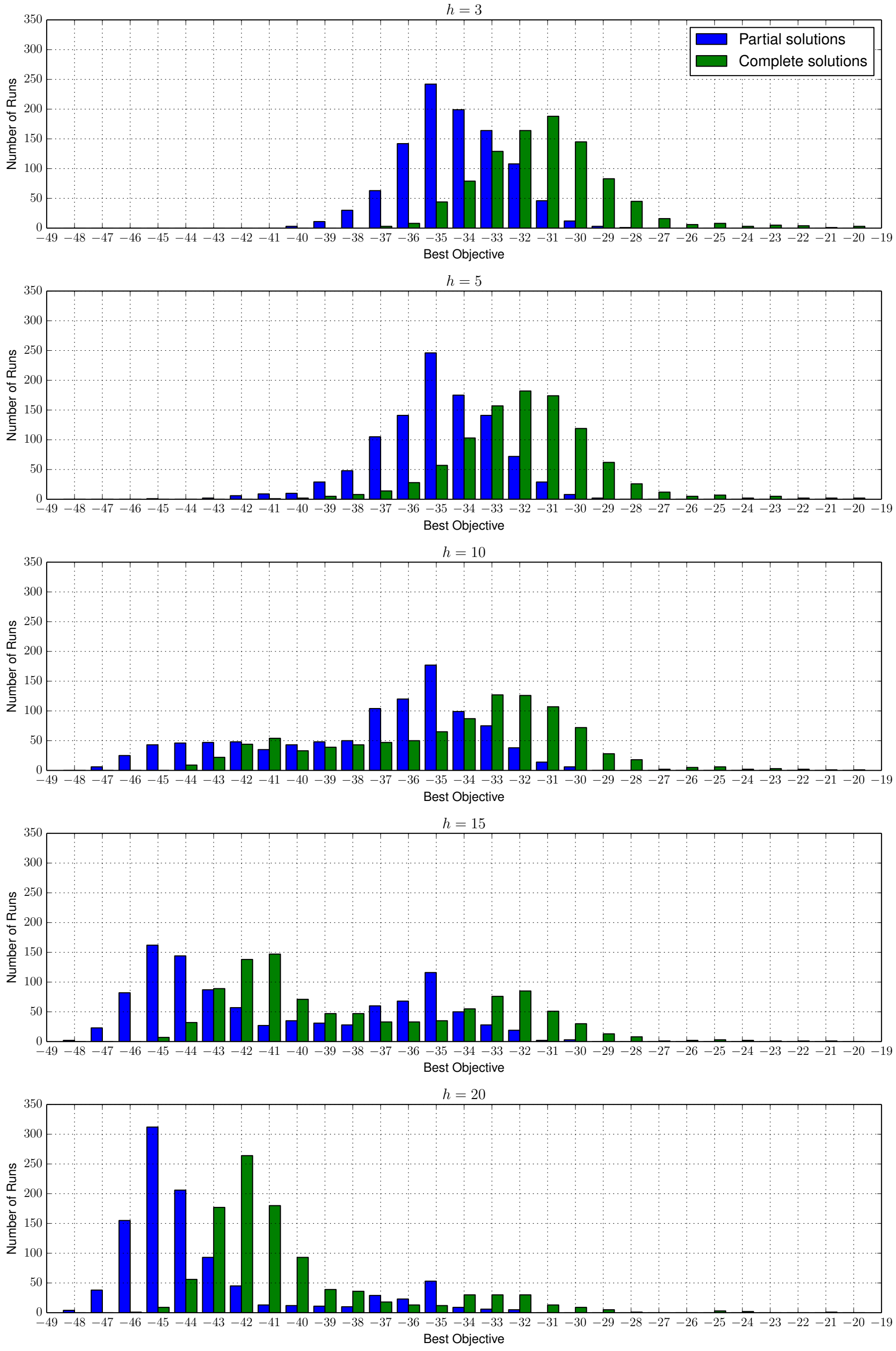


Figure 5.10: The results of the base case are re-run with dynamic neighborhood selection disabled for various static values of the restricted best-first neighborhood parameter h. The best solutions found are shown for each case.

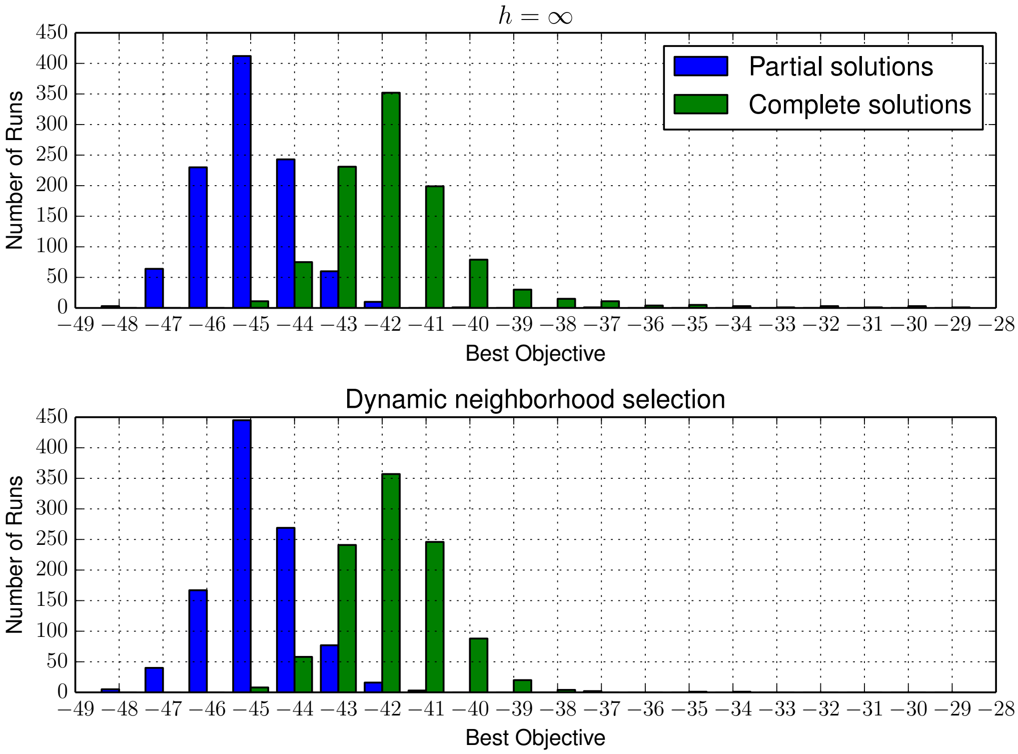


Figure 5.11: The best solutions found for the static neighborhood $h = \infty$ case and the dynamic neighborhood selection case.

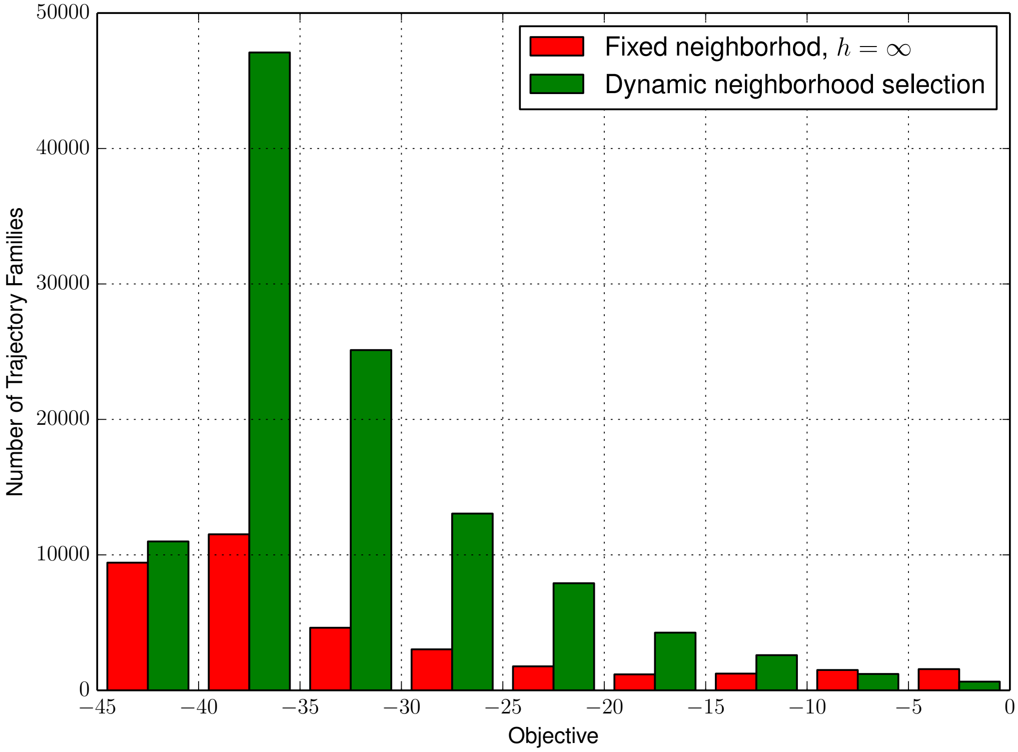


Figure 5.12: The number of trajectory families found for complete solutions in the dynamic neighborhood selection case and the static neighborhood $h = \infty$ case.

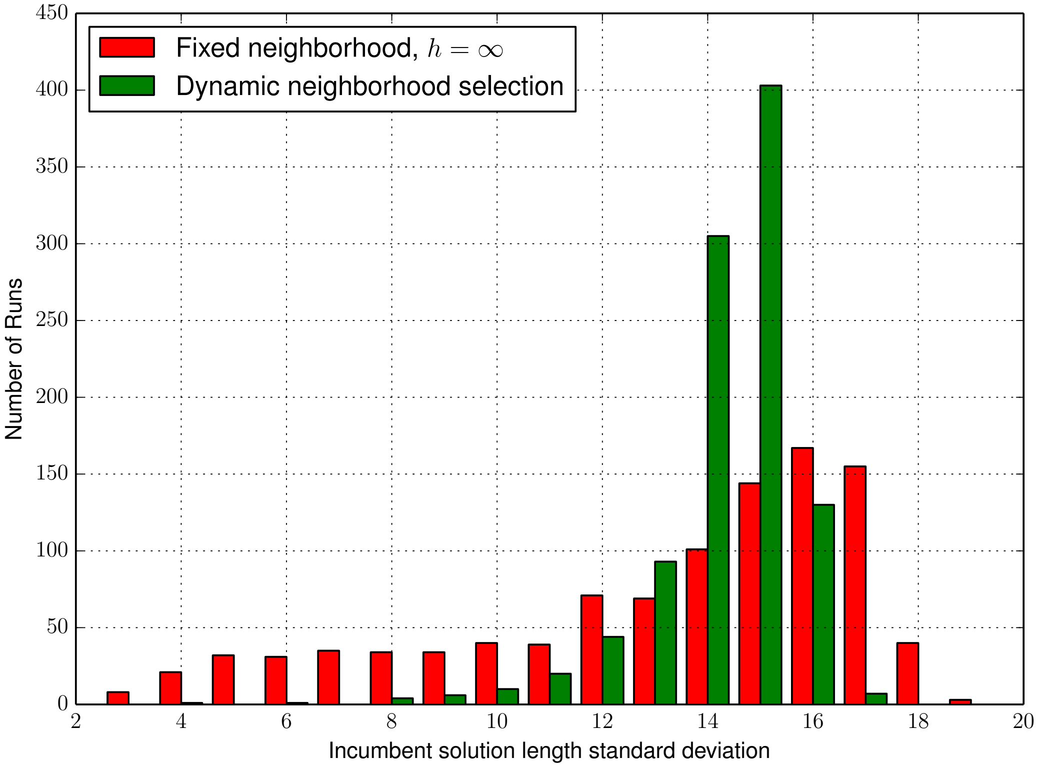


Figure 5.13: The standard deviation of the incumbent solution length for all runs in the dynamic neighborhood selection case and the static neighborhood $h = \infty$ case.

case as expected.

Although a static neighborhood can give good results, it does not give the breadth of solutions that dynamic neighborhood selection provides. Further, with $h = \infty$ the neighborhood size will monotonically increase as the search tree grows over the iterations. We would then expect to see a performance decrease over time compared to the dynamic neighborhood selection approach which restricts the neighborhood size. This scalability is important if we consider running the search over longer periods of time.

	Base Case	**Reduced Performance A** (5% change)	**Reduced Performance B** (10% change)
T_{max}	0.135 N	0.12825 N	0.1215 N
m_{dry}	500 kg	525 kg	550 kg

Table 5.10: Spacecraft parameters for base case and reduced spacecraft performance cases.

5.3.5 Reduced Spacecraft Performance

We now modify the spacecraft parameters for reduced performance. We consider a reduction to the spacecraft's maximum thrust magnitude T_{max} as well as to its available fuel mass. The reduction in thrust capability extends the duration of finite burn estimates for the impulsive maneuvers. The longer finite burn maneuvers consequently also experience greater gravity losses. The reduced fuel mass further limits the maneuverability of the spacecraft. The resulting solutions are more tightly constrained than those of the base case. We study this case both to examine the impact of reduced spacecraft performance on the quality of solutions found and to find more conservative solutions that are more likely to converge to low-thrust finite burn trajectories, noting that the finite burn feasibility constraints placed on impulsive solutions are based only on estimates of the corresponding finite burn maneuvers.

Table 5.10 gives the spacecraft parameters for the base case and two reduced performance cases. For the Reduced Performance A case, we decrease the spacecraft's maximum thrust magnitude by 5% and increase its dry mass by 5% (keeping the initial mass constant and therefore reducing the available fuel mass). For the Reduced Performance B case we instead change the values by 10% for even more conservative results. We generate results for the reduced performance cases using the base case parameters shown in Table 5.6 with the new reduced spacecraft performance parameters. Figure 5.14 shows the best solutions for these cases (over 1024

	Minimum	**Median**	**Maximum**
Partial solutions			
Base case	-48	-45	-37
Reduced performance A	-47	-44	-40
Reduced performance B	-46	-43	-37
Complete solutions			
Base case	-45	-42	-34
Reduced performance A	-45	-42	-34
Reduced performance B	-43	-40	-32

Table 5.11: Summary of best solutions found for base case and reduced spacecraft performance cases.

runs) in comparison to the base case.

We see the reduced performance cases have similar results to the base case. However, the objective distributions have shifted. For partial solutions, the median objective value increases by one for each 5% reduction in the spacecraft's performance. For complete solutions, the Reduced Performance A case achieves the same minimum, median and maximum objective, while the values for the Reduced Performance B case are increased by two. Table 5.11 summarizes the differences in the solution statistics. We continue by investigating individual runs from both reduced performance cases.

Reduced Performance Case A

We examine a single run from the collection of 1024 runs of Reduced Performance Case A. We consider run 622 corresponding to a launch date of 59466.01 MJD and a best complete solution objective of $J = -45$. The run's iteration history is shown in Figure 5.15. The best complete solution objective of $J = -45$ is achieved

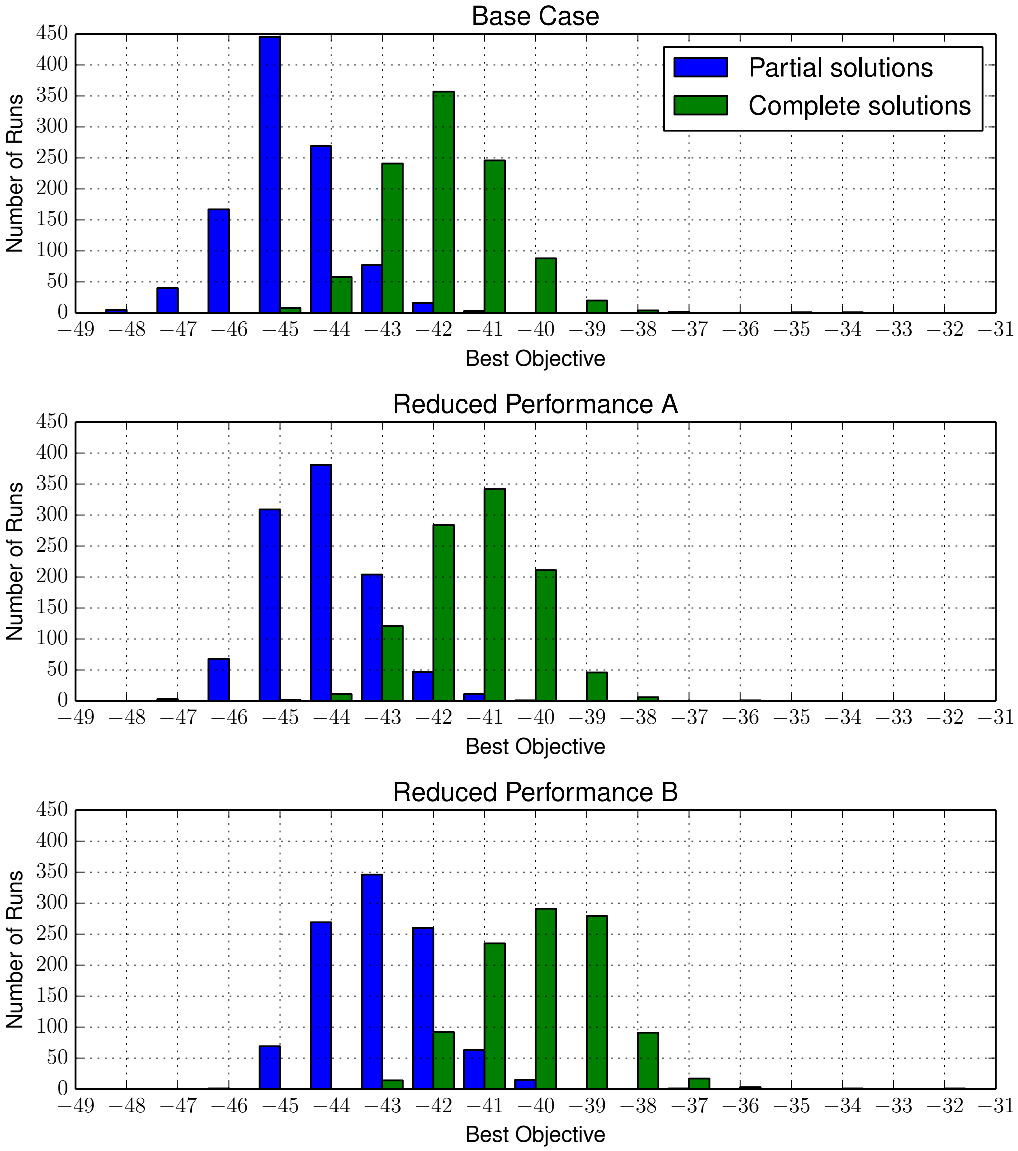
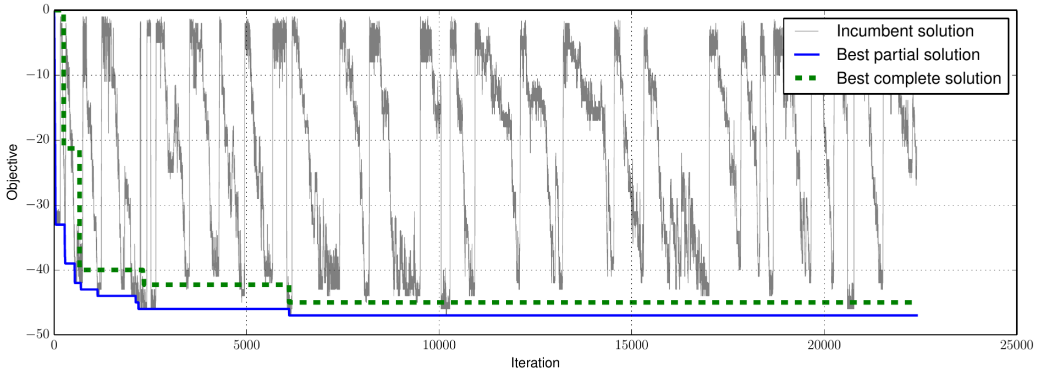
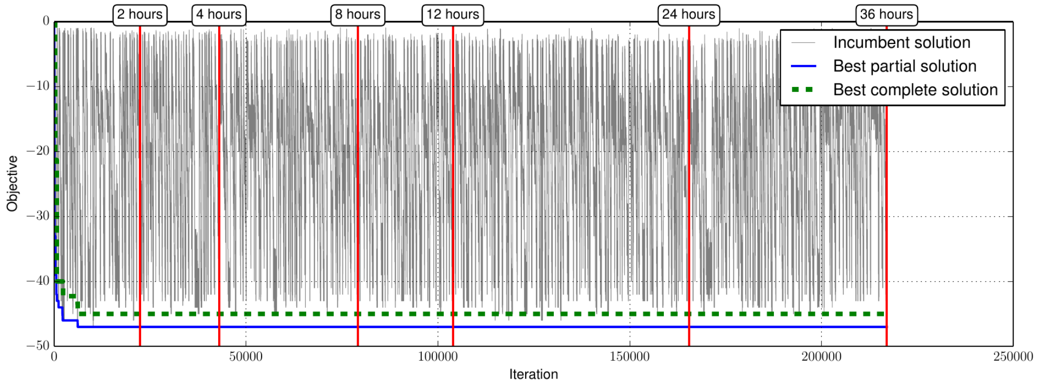


Figure 5.14: Reduced spacecraft performance cases compared to base case: Best partial and complete solutions generated by 1024 runs over launch dates from 2015 to 2025.



(a) 2 hour run time.



(b) 36 hour run time.

Figure 5.15: Reduced spacecraft performance case A: run #622/1024. The objective history for the incumbent solution and best found partial and complete solutions are shown for (a) 2 hour run time and (b) 36 hour run time.

no.	t (MJD)	mass (kg)	Asteroid	no.	t (MJD)	mass (kg)	Asteroid
0	59466.01	1500.00	Earth	1	59521.45	1500.00	2007VL3
2	59646.51	1475.03	2005CD69	3	59712.87	1445.24	164207
4	59768.40	1425.32	2003LN6	5	59856.87	1419.93	2001SQ3
6	59963.48	1387.41	2003YG136	7	60051.66	1349.72	2002GR
8	60132.93	1328.15	186844	9	60248.09	1296.64	2006UB17
10	60340.42	1274.68	2002AU4	11	60421.69	1255.34	2005EB30
12	60513.76	1224.11	163364	13	60561.85	1205.37	2004FD
14	60657.28	1188.57	2008GF1	15	60778.42	1157.35	2008PW4
16	60852.23	1127.47	138175	17	60925.93	1119.86	2002EM7
18	61014.46	1089.48	2000SZ162	19	61161.70	1054.68	2005EZ169
20	61217.45	1050.13	2008JP24	21	61287.36	1034.04	2000AA6
22	61350.13	1013.53	1997UA11	23	61416.64	991.23	2006KV89
24	61508.61	974.09	2000EB14	25	61586.09	947.95	2004JO20
26	61685.62	924.96	2003OT13	27	61774.17	887.88	2006VU2
28	61837.15	878.08	2006XP4	29	61940.03	858.12	2006GC1
30	62020.65	839.36	2003XK	31	62104.66	798.21	2006RJ1
32	62171.01	780.54	1998UY24	33	62241.09	759.72	175706
34	62311.30	743.73	2008CP	35	62367.10	725.61	2006HF6
36	62429.73	712.75	2008LG2	37	62481.67	686.30	2004PR92
38	62537.42	678.12	2001RQ17	39	62611.49	661.37	2001XG1
40	62700.21	645.89	2004BN41	41	62792.83	617.95	2004UT1
42	62844.78	600.03	2008AP33	43	62886.17	588.39	2006QK33
44	62941.61	576.69	2002TX59	45	62993.56	556.94	2006VY2
46	63117.74	525.32	2006QQ56				

Table 5.12: Reduced performance case A: run #622/1024. The tour itinerary is shown for a $J = -45$ solution rendezvousing with asteroid 2006QQ56.

early within the two hour run time at iteration 6100 of 22401. We extend this run to compute for 36 hours–however we see no further improvement for partial or complete solutions.

We examine a single complete solution from this run for $J = -45$. We choose a trajectory rendezvousing with asteroid 2006QQ56. Figure 5.16 shows the trajectory, and Table 5.12 shows the tour itinerary. The spacecraft mass history closely follows the linear budget as in the other cases. Since this trajectory uses more conservative values for the spacecraft performance, we expect that this solution

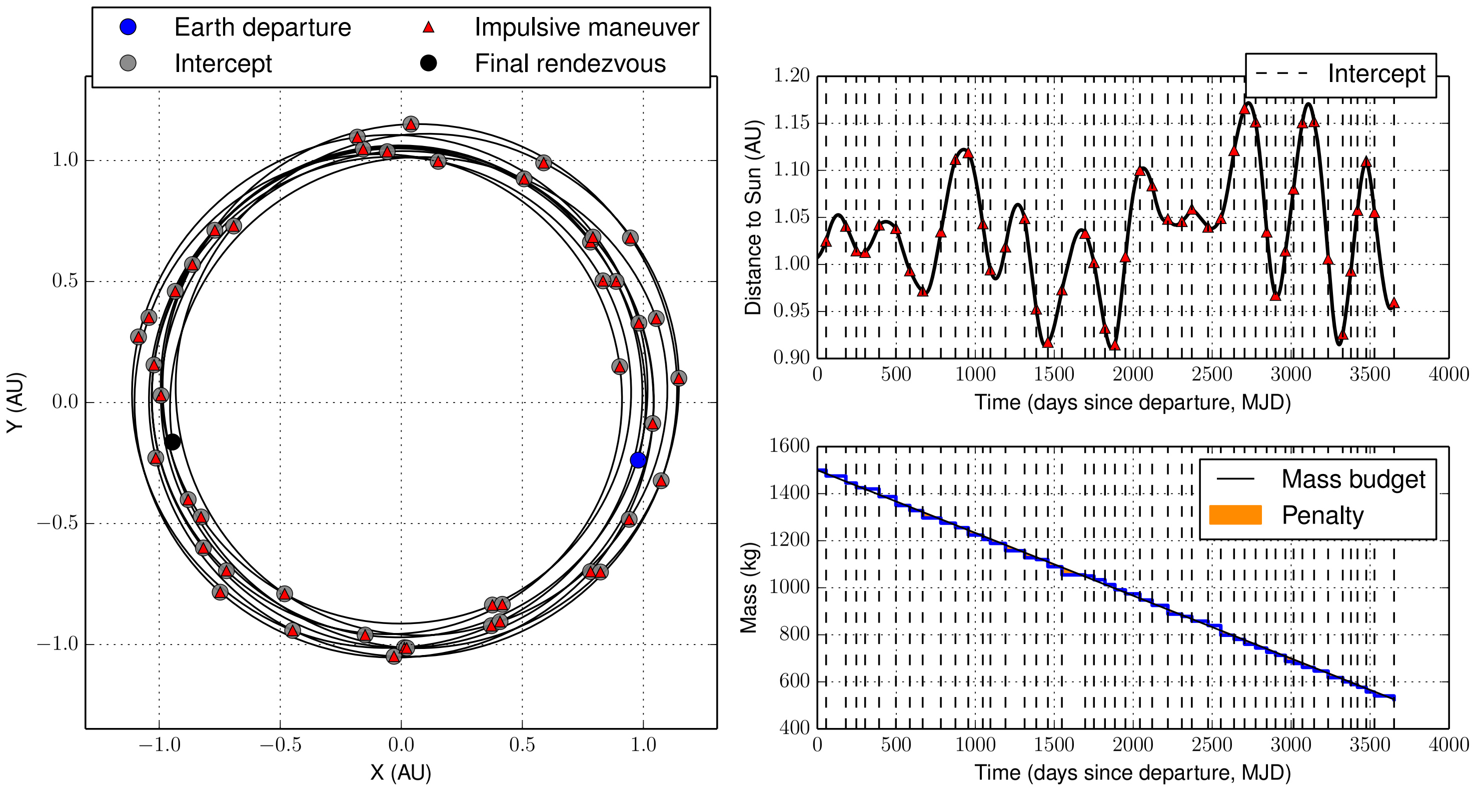
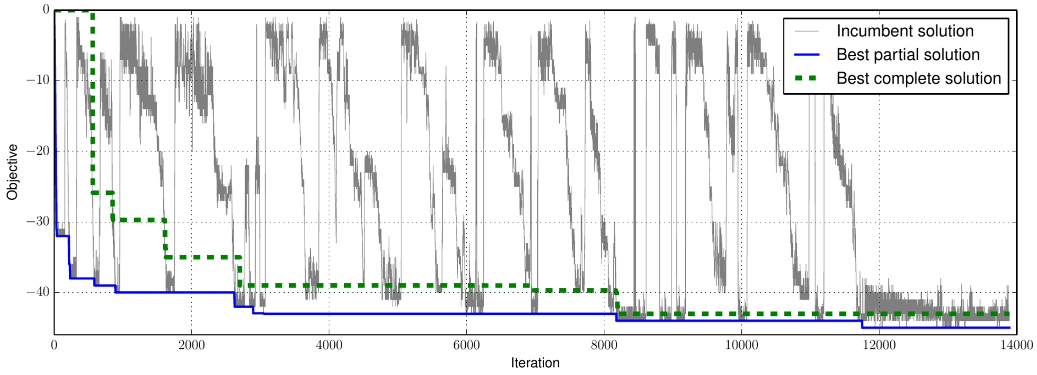
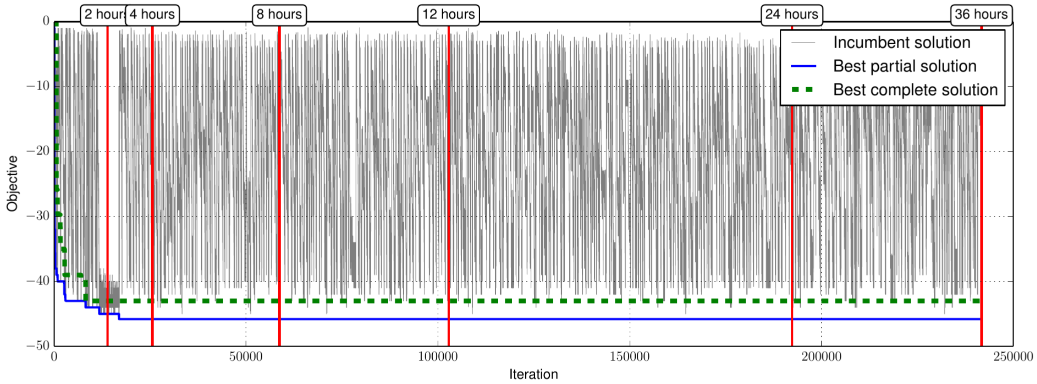


Figure 5.16: Reduced performance case A: run #622/1024. The trajectory is shown for a $J = -45$ solution rendezvousing with asteroid 2006QQ56.



(a) 2 hour run time.



(b) 36 hour run time.

Figure 5.17: Reduced spacecraft performance case B: run #402/1024. The objective history for the incumbent solution and best found partial and complete solutions are shown for (a) 2 hour run time and (b) 36 hour run time.

could more easily be converted into a feasible low-thrust finite burn trajectory.

Reduced Performance Case B

We similarly examine a single run from the collection of 1024 runs of Reduced Performance Case B. We consider run 402 corresponding to a launch date of 58601.92 MJD and a best complete solution objective of $J = -43$. The run's iteration history is shown in Figure 5.17. The best complete solution objective of $J = -43$ is achieved at iteration 8196 of the total 13892 iterations for the two hour run time. Again, we extend the search to 36 hours to see if better solutions can be found. We see that the search does not provide any improved complete solutions with the additional runtime. The best partial solution objective improves to $J = -46$ before 4 hours has elapsed, but improves no further afterward.

We examine a single complete solution from this run for $J = -43$. We choose a trajectory rendezvousing with asteroid 2006UB17. Figure 5.18 shows the trajectory, and Table 5.13 shows the tour itinerary. The spacecraft mass history closely follows the linear budget as in the other cases. Again, since this trajectory uses more conservative values for the spacecraft performance, we expect that this solution could more easily be converted into a feasible low-thrust finite burn trajectory.

For both runs of the reduced performance cases, we see that the search finds its best complete solutions within two hours of run time, even when the searches are continued for up to 36 hours. Thus, the search algorithm appears to find good solutions rapidly, with little improvement for extended run times.

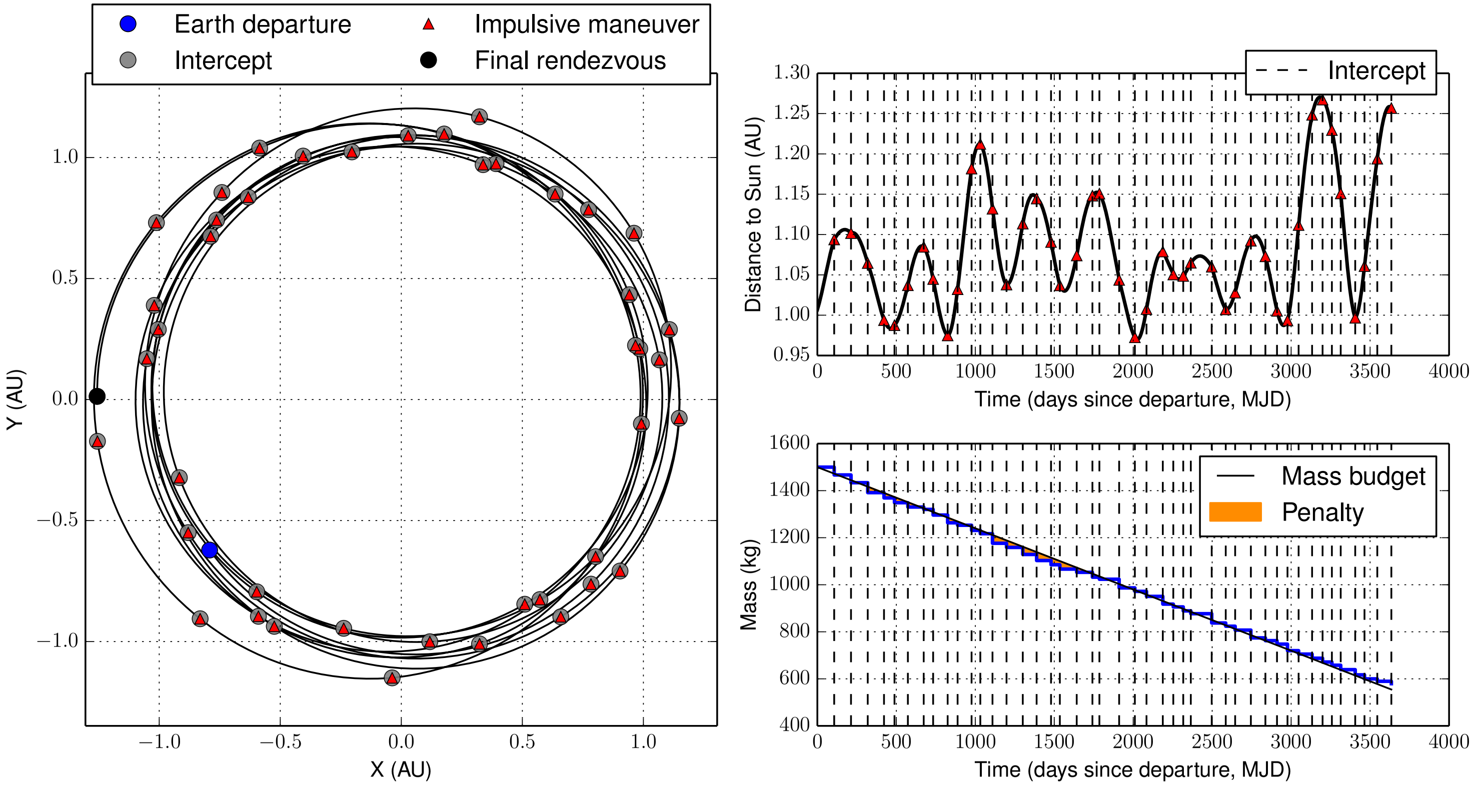


Figure 5.18: Reduced spacecraft performance case B: run #402/1024. The trajectory is shown for a $J = -43$ solution rendezvousing with asteroid 2006UB17.

no.	t (MJD)	mass (kg)	Asteroid	no.	t (MJD)	mass (kg)	Asteroid
0	58601.92	1500.00	Earth	1	58708.53	1500.00	2000QW7
2	58815.02	1466.78	2003YT70	3	58920.94	1433.92	2005EZ169
4	59023.98	1391.65	2007LF	5	59090.33	1369.23	175729
6	59175.11	1348.86	2004UT1	7	59275.37	1330.43	2004XG
8	59335.02	1320.35	2005VO	9	59426.99	1296.31	2004MS1
10	59489.99	1263.38	2005WK4	11	59578.47	1252.98	1999VM11
12	59633.79	1231.10	2004BF11	13	59710.95	1217.08	152563
14	59799.78	1176.37	2007DS84	15	59902.66	1158.79	2003OQ13
16	59990.32	1128.78	2003WY153	17	60081.63	1103.00	2008JP24
18	60137.44	1085.03	2008CP	19	60244.17	1066.57	2008WM
20	60343.70	1052.84	2008CY118	21	60387.79	1032.13	2001TB
22	60512.86	1023.40	2007YM	23	60612.39	986.64	2001SE270
24	60686.09	971.82	2005YP180	25	60789.33	950.53	2006SD25
26	60855.68	917.67	2006WG130	27	60918.69	905.75	2006BY8
28	60966.67	888.86	2002VX91	29	61098.95	876.94	2004FD
30	61187.78	837.46	2008TC3	31	61247.44	823.68	2005GC120
32	61346.97	807.12	2007CC27	33	61438.93	774.13	7335
34	61512.46	762.33	2008CN116	35	61578.82	747.57	2004RQ252
36	61649.26	719.79	2006BZ147	37	61734.64	705.10	1995CR
38	61800.99	687.61	2008EE9	39	61860.01	671.39	2007DB83
40	61915.81	657.86	2008UB7	41	62007.66	639.05	2003SW130
42	62066.38	616.91	2004JN1	43	62147.41	599.79	2005YK
44	62236.48	579.46	2006UB17				

Table 5.13: Reduced spacecraft performance case B: run #402/1024. The tour itinerary is shown for a $J = -43$ solution rendezvousing with asteroid 2006UB17.

5.3.6 Comparison to GTOC4 Winning Solution

In the previous sections we generated solutions across the full 10-year range of allowed launch epochs (Table 5.5) and found solutions that compared favorably to the overall GTOC4 competition results. In this section we instead wish to more directly compare against the winning GTOC4 solution found by Moscow State University [11]. In their solution, the spacecraft departs the Earth at 58676.40 MJD and visits 44 intermediate asteroids before completing a final rendezvous with asteroid 2000SZ162, achieving an objective value of $J = -44$. We now examine solutions for the same launch epoch, and place emphasis on solutions that similarly rendezvous with asteroid 2000SZ162.

We first generate solutions for the base case parameters given in Table 5.6 for the launch epoch of 58676.40 MJD and allow the search to run for 12 hours. Figure 5.19 shows the number of trajectory families found and their objective values for complete solutions. The best overall solution is found after 12 hours and rendezvouses with asteroid 2008JP24 for an objective value of $J = -43$. The best solution rendezvousing with asteroid 2000SZ162 is found after 4 hours and visits 41 intermediate asteroids, fewer than the 44 achieved by the winning GTOC4 solution. The $J = -41$ objective achieved is close to the median of $J = -42$ for all runs of the base case (Table 5.7). Although a $J = -44$ solution rendezvousing with asteroid 2000SZ162 was not found for the base case parameters at this launch epoch, we note that the base case executed across the full range of launch epochs did find such solutions (Table 5.8). These were found at the nearby launch epochs of 58032.41 MJD, 58323.06 MJD, 58448.74 MJD and 58955.41 MJD.

The base case solutions found are constrained so that finite burn estimates for the impulsive maneuvers do not overlap in time in an effort to ensure that the resulting impulsive trajectories also have valid finite burn representations. These constraints are only estimates, however, and are specific to the impulsive tour model

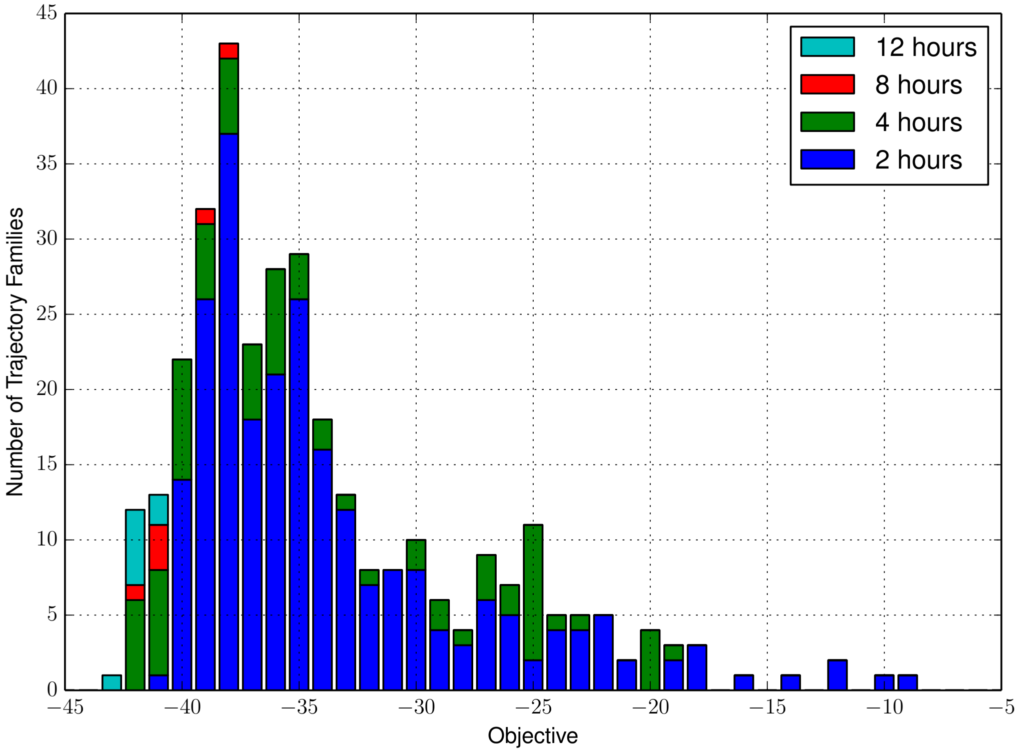


Figure 5.19: Number of trajectory families found versus objective for the base case parameters and a launch epoch of 58676.40 MJD. The search is allowed to run for 12 hours. Only complete solutions are shown.

used in this work. Further, these constraints do not guarantee that a feasible conversion is possible and may instead over- or under-constrain the search. Section 5.3.2 showed that the constraints significantly reduce the feasible solution space, in turn limiting the objectives achievable. We note that the Moscow State University search methodology also operates on impulsive trajectories, but allows finite burn estimates for impulsive maneuvers to overlap in time [29]. When the finite burn conversion occurs, many of these overlaps are resolved in the optimization, while some asteroids are dropped from the itinerary as necessary. It therefore makes sense to consider relaxing (but not eliminating) the finite burn constraints used in our model.

Increased Spacecraft Performance

We now consider relaxing the finite burn constraints through an increase to the maximum allowed thrust T_{max} of the spacecraft. The increased T_{max} reduces the duration of finite burn estimates for the impulsive maneuvers, and therefore makes the finite burn constraints less restrictive. One can equivalently view this as allowing an overlap in finite burn estimates for the original T_{max} value. We generate solutions for the base case parameters again for a launch epoch of 58676.40 MJD, but increase the value of T_{max} by 5% from 0.135 N to 0.14175 N. Figure 5.20 shows the distribution of trajectory families found for complete solutions up to a 12 hour runtime. The best solutions are found after 4 hours and achieve an objective of $J = -46$, exceeding the best results of the base case previously executed across the full 10-year launch window. These best three trajectory families rendezvous with two different final asteroids, 2006DN and 2003YG136. Ten trajectory families rendezvousing with seven final asteroids are found with objective values of $J = -45$. One of these rendezvouses with asteroid 2000SZ162. We have found a solution rendezvousing with the same final asteroid for the same launch epoch that improves upon the winning GTOC4 solution. Figure 5.21 shows the trajectory, and Table 5.14

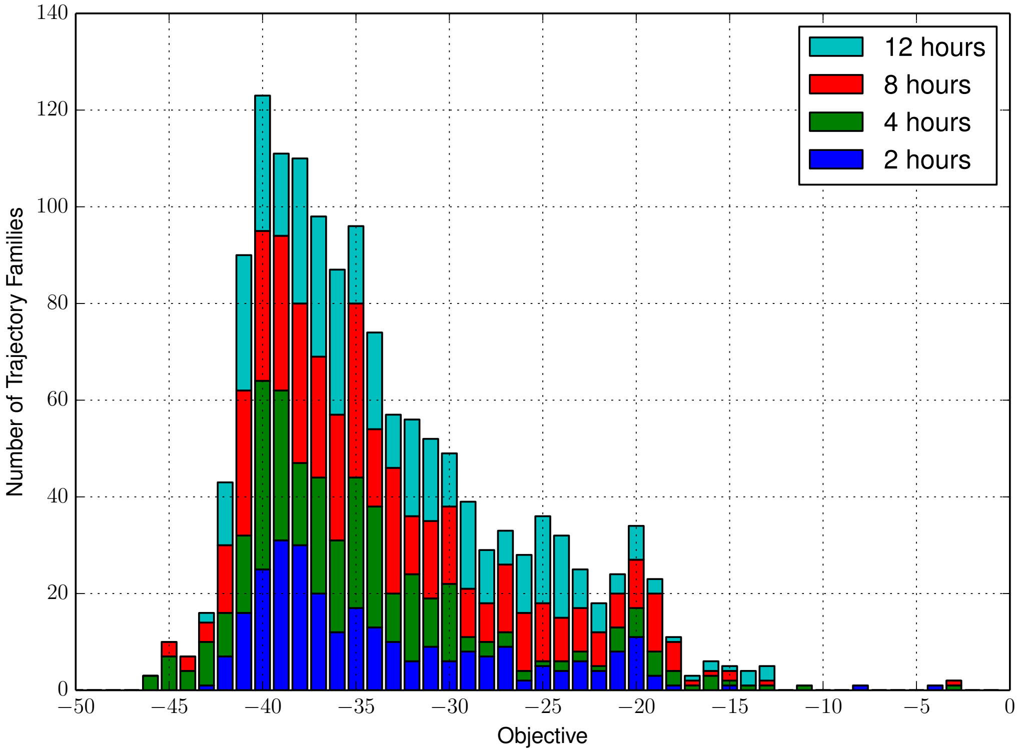


Figure 5.20: Number of trajectory families found versus objective for the increased spacecraft performance case and a launch epoch of 58676.40 MJD. The spacecraft's maximum allowed thrust T_{max} is increased by 5% from 0.135 N to 0.14175 N. The search is allowed to run for 12 hours. Only complete solutions are shown.

no.	t (MJD)	mass (kg)	Asteroid	no.	t (MJD)	mass (kg)	Asteroid
0	58676.40	1500.00	Earth	1	58739.41	1500.00	2000QW7
2	58845.33	1482.62	2006XP4	3	58970.38	1413.66	2006UJ185
4	59091.90	1390.88	2005NW44	5	59140.15	1367.49	2007TK15
6	59210.41	1360.11	2005VY1	7	59280.62	1320.97	199003
8	59336.37	1303.26	2005VO	9	59529.95	1281.63	2000RN77
10	59606.66	1258.96	2003SW130	11	59735.44	1239.92	2007CC27
12	59805.52	1199.20	2005TF45	13	59890.90	1191.00	5496
14	59964.43	1161.82	1999YR14	15	60012.52	1151.97	2008SW150
16	60096.53	1137.19	1998DK36	17	60155.42	1125.36	2003GD
18	60210.68	1106.51	2002JR100	19	60339.53	1092.84	2004RN111
20	60423.54	1053.52	2006UQ216	21	60497.60	1034.50	1995CR
22	60541.84	1009.89	2008GF1	23	60586.08	1000.45	7335
24	60678.40	993.92	2003QY29	25	60770.68	952.51	2007VD8
26	60848.16	927.45	136849	27	60896.25	895.36	2002CX58
28	60980.65	888.44	2006KC40	29	61054.72	860.19	2003NO4
30	61110.25	845.91	2006RJ1	31	61198.59	838.29	2007VV6
32	61257.61	819.55	2001BA16	33	61338.88	797.59	2006KZ39
34	61412.58	778.92	2002NW	35	61497.63	762.74	2004SA20
36	61571.70	732.29	2007XO	37	61626.86	714.09	2007YF
38	61656.69	707.29	2007CR5	39	61726.09	692.93	2002VX91
40	61806.89	664.78	2006QQ56	41	61851.09	648.01	2008GM2
42	61954.41	628.94	2005UF1	43	62031.89	606.11	2007YM
44	62116.51	577.94	2006UP217	45	62182.25	568.56	2007RQ12
46	62312.33	516.35	2000SZ162				

Table 5.14: Increased spacecraft performance case: The tour itinerary is shown for a $J = -45$ solution rendezvousing with asteroid 2000SZ162.

shows the tour itinerary. We do note that the sequence of intermediate asteroids in this solution is different than that of the winning GTOC4 solution. In our results we were unable to reproduce that solution's exact sequence. We have however found a comparable trajectory for the same launch epoch and rendezvous asteroid, as well as many other solutions with the same or superior objective values. Due to the large asteroid population and the vast number of feasible sequences, we find it unlikely that any two methodologies will reproduce the same asteroid sequence in reasonable compute times. This is in part due to the differing constraints, objec-

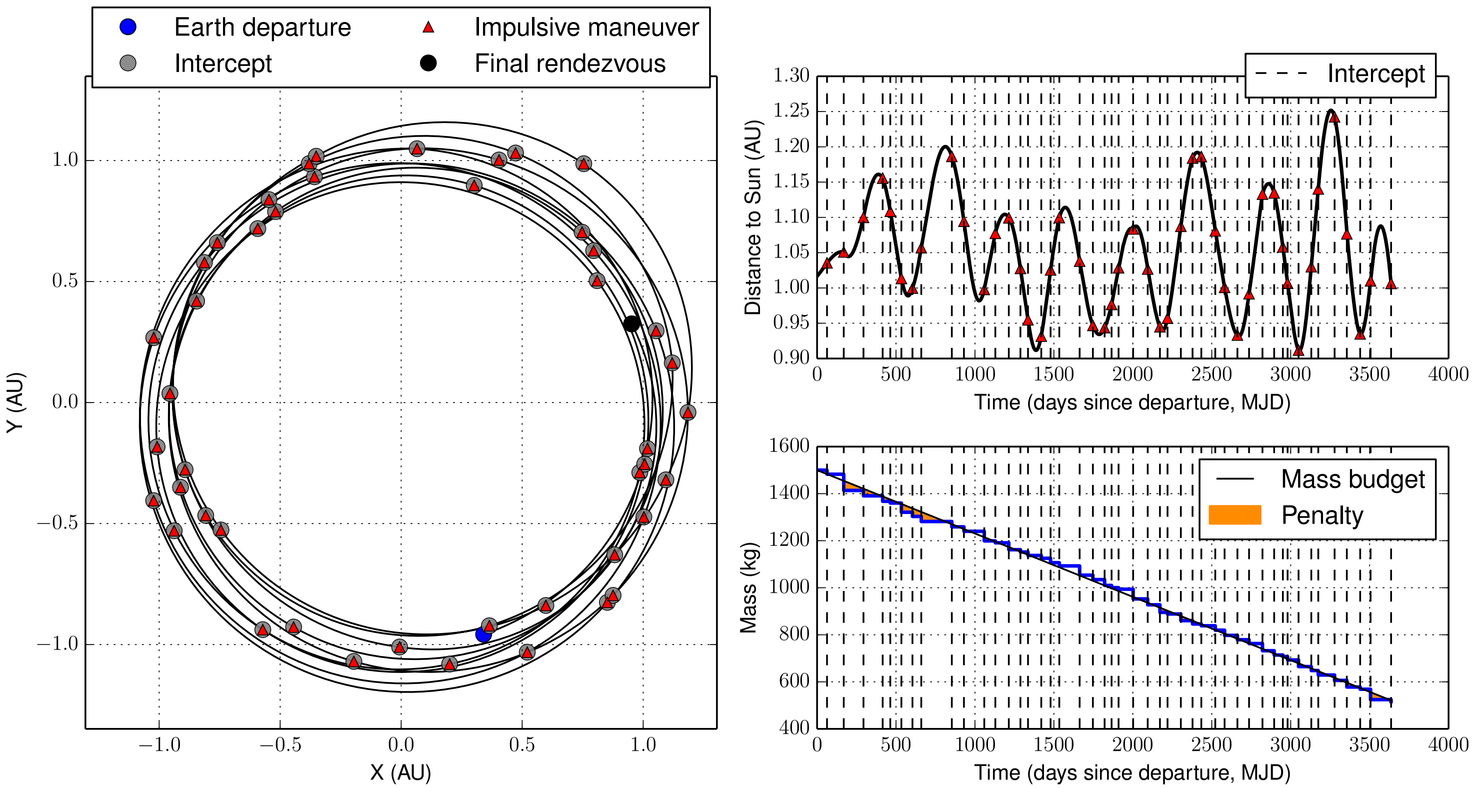


Figure 5.21: Increased spacecraft performance case: The trajectory is shown for a $J = -45$ solution rendezvousing with asteroid 2000SZ162.

tives and penalties placed on solutions during the search, and is supported by the original GTOC4 results where no two asteroid sequences were replicated. Further, we note that the top two solutions had similar launch epochs (within two weeks of each other) and rendezvoused with the same final asteroid, but differed significantly in the intermediate asteroid sequence, sharing only three asteroids in common.

Relaxing the finite burn constraints through an increase to the spacecraft's maximum allowed thrust magnitude T_{max} increases the variety and quality of solutions found and allows us to find solutions comparable to the winning GTOC4 solution. We note again that these are impulsive solutions, however, and that there is a trade-off: with less restrictive finite burn constraints, we expect the subsequent finite burn conversion to be more difficult. The next section considers the low-thrust finite burn conversion of selected impulsive solutions.

5.3.7 Low-thrust Finite-burn Conversion

The GTOC4 problem statement requires low-thrust finite burn trajectories that adhere to the specified maximum thrust T_{max}. The solutions we have presented up to this point have all been impulsive, ignoring this limit on the thrust magnitude. We have however constrained the impulsive solutions based on estimates of representative finite burn maneuvers; these constraints further limit the feasible solution space in an effort to only find trajectories that can be converted to low-thrust finite burn solutions. These finite burn estimates are only approximations, however, and we are not guaranteed that our solutions will converge to valid finite burn trajectories. This section evaluates the finite burn conversion and optimization of selected impulsive solutions. The resulting low-thrust finite burn trajectories can be compared directly to the best known solutions of GTOC4 (Section 5.2).

An impulsive solution generated by the search algorithm can act as an initial guess for a low-thrust finite burn trajectory. In this conversion, the sequence of target asteroids of the impulsive solution becomes fixed. The impulsive maneuvers are then replaced with finite burn maneuvers that must satisfy the original problem constraints. The resulting problem is a continuous optimization problem with no discrete decision variables. We can solve the problem in a variety of ways, either formulating it as a parameter optimization problem or instead taking an optimal control approach, for example. Existing tools such as Copernicus are well suited to this task [71]. Olympio has developed an optimal control formulation that can optimize GTOC4 trajectories directly [52]. The method takes an initial guess in the form of the asteroid sequence, asteroid visit times, and spacecraft velocity and mass values along the trajectory and generates mass-optimal low-thrust finite burn trajectories. We use Olympio's approach and software to find finite burn solutions based on our impulsive solutions. The optimization varies the thrust history of the spacecraft, but in our usage treats the asteroid encounter dates as fixed. The

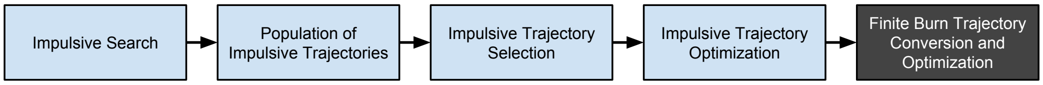

Impulsive Search
Population of Impulsive Trajectories
Impulsive Trajectory Selection
Impulsive Trajectory Optimization
Finite Burn Trajectory Conversion and Optimization

Figure 5.22: Workflow for GTOC4 solution process. The impulsive search algorithm generates a population of impulsive trajectories. The user then selects specific impulsive trajectories, optimizes them impulsively, and passes them to the black-box finite burn conversion and optimization tool.

resulting solutions are optimal for the fixed encounter dates, but we expect they would be further improved if the encounter dates were also free to vary.

Reduced Performance Case A

We first consider the $J = -45$ impulsive trajectory from Reduced Performance Case A (Figure 5.16 and Table 5.12). This solution was generated for a 5% reduction in the spacecraft's maximum thrust magnitude and a 5% increase in its dry mass.

We generate the initial guess using the search-generated impulsive solution and run the optimization using Olympio's approach. In this case, we are not initially able to find a feasible finite burn solution satisfying the problem constraints. A limitation of our usage of Olympio's approach is that the asteroid encounter dates are fixed–in general these encounter dates could also be optimized, or in this case modified to find a feasible solution. We now consider the additional step of optimizing the search-generated impulsive trajectory to maximize the final mass, noting that the search-generated solutions are not optimal in part because the encounter dates are chosen from a discretization of possible dates. We use the same constraints as in the search; however we now consider the asteroid sequence to be fixed, and only optimize the launch date and encounter dates. The workflow for the full solution process is shown in Figure 5.22.

We find that optimizing the launch and encounter dates for the impulsive trajectory increases the final mass from 525.32 kg to 643.86 kg. The optimized

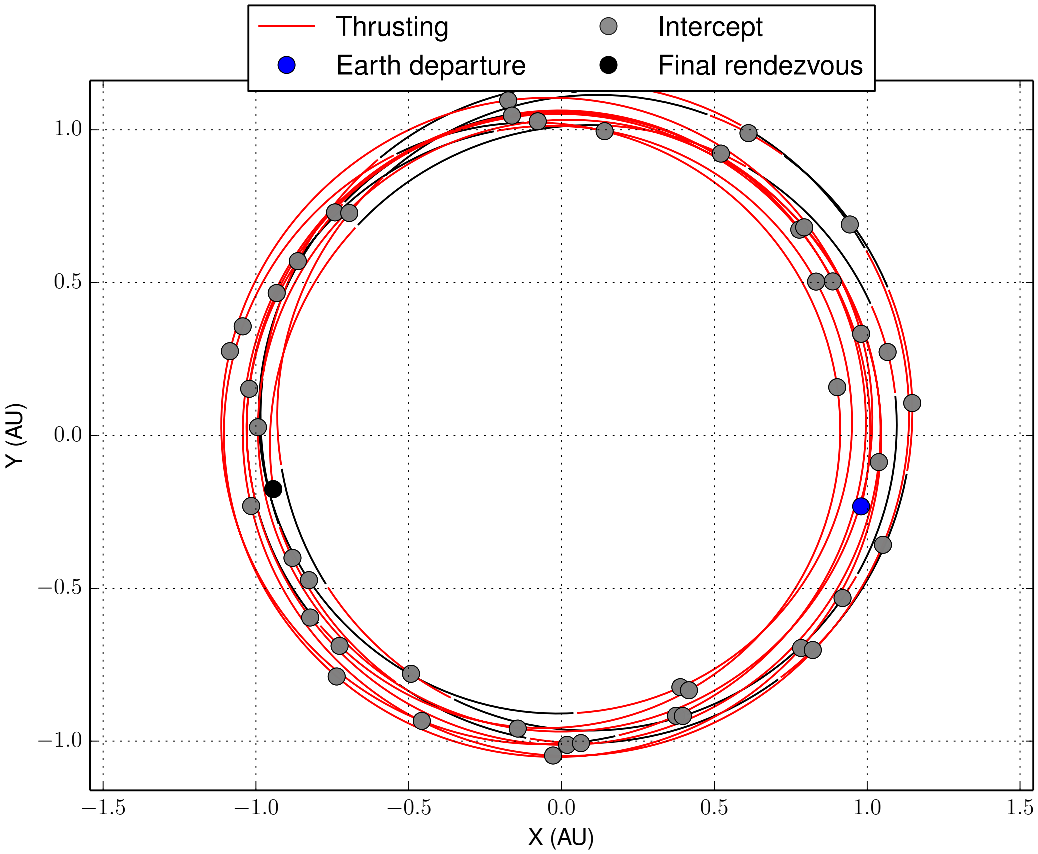


Figure 5.23: Low-thrust finite burn trajectory corresponding to the $J = -45$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.16 and Table 5.12.

impulsive trajectory still adheres to the original model constraints. We now optimize this solution using Olympio's approach, and find that it does converge to an optimal low-thrust finite burn trajectory. Figures 5.23 and 5.24 show the solution, and Table 5.15 shows the tour itinerary. The final mass of the trajectory is 505.12 kg, just above the limit of 500 kg. This solution exceeds the best known GTOC4 solution of $J = -44$.

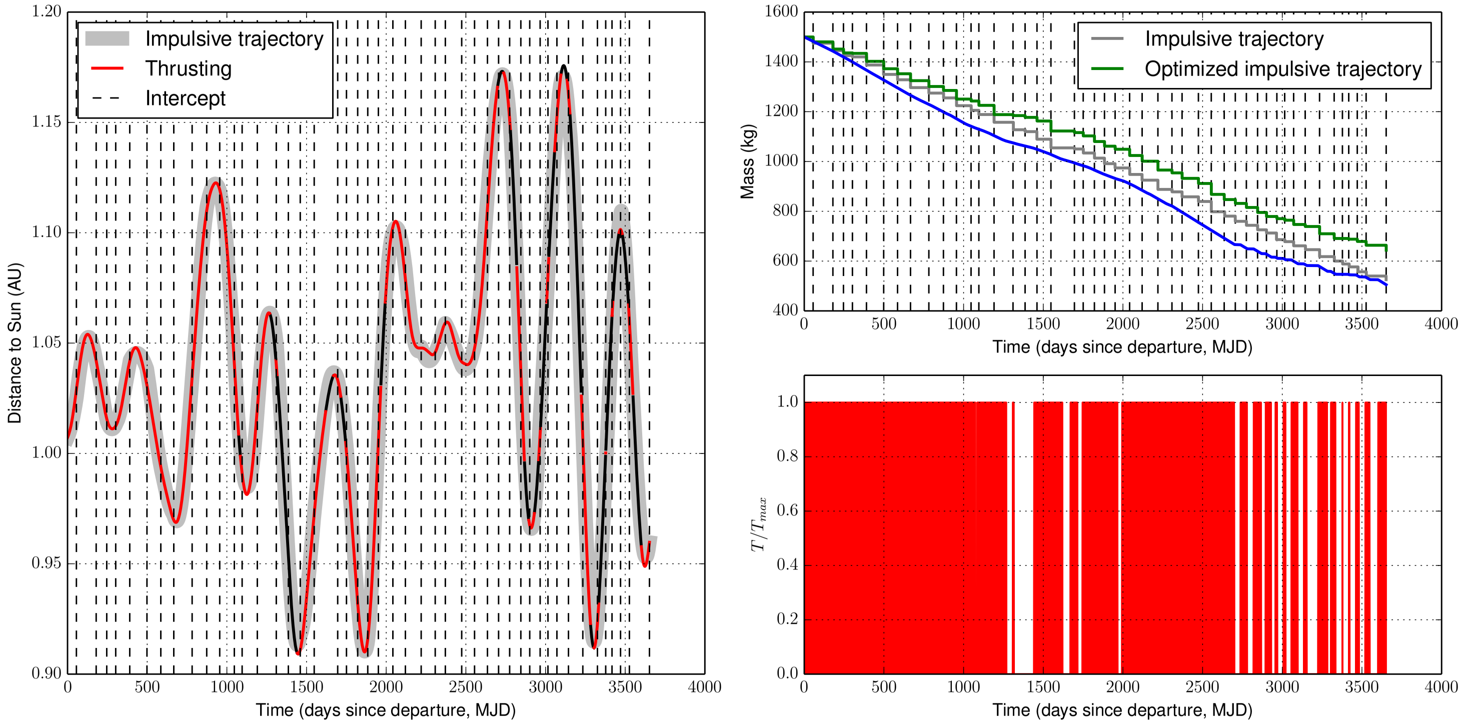


Figure 5.24: Low-thrust finite burn trajectory corresponding to the $J = -45$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.16 and Table 5.12. The original impulsive trajectory is also shown.

no.	t (MJD)	mass (kg)	Asteroid	no.	t (MJD)	mass (kg)	Asteroid
0	59466.35	1500.00	Earth	1	59522.01	1482.87	2007VL3
2	59646.13	1443.13	2005CD69	3	59712.44	1419.88	164207
4	59768.76	1399.81	2003LN6	5	59856.82	1367.21	2001SQ3
6	59964.82	1326.88	2003YG136	7	60051.80	1294.77	2002GR
8	60133.21	1266.10	186844	9	60248.35	1229.74	2006UB17
10	60339.88	1198.87	2002AU4	11	60421.47	1171.73	2005EB30
12	60513.15	1141.77	163364	13	60562.14	1130.04	2004FD
14	60657.46	1103.90	2008GF1	15	60776.49	1074.82	2008PW4
16	60852.29	1061.95	138175	17	60926.93	1048.99	2002EM7
18	61014.49	1027.92	2000SZ162	19	61161.87	994.24	2005EZ169
20	61217.04	983.12	2008JP24	21	61287.33	965.70	2000AA6
22	61350.55	947.92	1997UA11	23	61417.28	932.02	2006KV89
24	61508.23	910.54	2000EB14	25	61586.83	884.05	2004JO20
26	61685.83	850.00	2003OT13	27	61774.00	821.49	2006VU2
28	61837.49	796.35	2006XP4	29	61940.14	755.61	2006GC1
30	62020.78	723.64	2003XK	31	62101.87	691.49	2006RJ1
32	62171.65	666.46	1998UY24	33	62241.05	651.05	175706
34	62311.32	639.17	2008CP	35	62367.09	627.78	2006HF6
36	62432.42	612.26	2008LG2	37	62481.55	607.35	2004PR92
38	62537.71	599.40	2001RQ17	39	62610.26	585.80	2001XG1
40	62700.35	578.14	2004BN41	41	62792.01	550.38	2004UT1
42	62843.90	546.90	2008AP33	43	62883.52	546.75	2006QK33
44	62937.09	541.34	2002TX59	45	62992.59	534.42	2006VY2
46	63118.46	505.12	2006QQ56				

Table 5.15: Low-thrust finite burn trajectory corresponding to the $J = -45$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.16 and Table 5.12. The tour itinerary is shown.

Reduced Performance Case B

We now consider the $J = -43$ impulsive trajectory from Reduced Performance Case B (Figure 5.18 and Table 5.13). This solution was generated for a 10% reduction in the spacecraft's maximum thrust magnitude and a 10% increase in its dry mass.

We again follow the workflow shown in Figure 5.22. Optimizing the launch and encounter dates for the impulsive trajectory increases the final mass from 579.46 kg to 679.11 kg. We now optimize this solution using Olympio's approach, and find that it does converge to an optimal low-thrust finite burn trajectory. Figure 5.25 and 5.26 show the solution, and Table 5.16 shows the tour itinerary. The final mass of the trajectory is 583.91 kg, well above the limit of 500 kg. This solution would have placed third in the original GTOC4 competition.

We found that both of the impulsive trajectories from both reduced performance cases could be converted to optimal low-thrust finite burn solutions. In these cases we added an intermediate impulsive optimization to maximize the final mass. The low-thrust finite burn trajectory corresponding to the first case represents the best known solution to the GTOC4 problem. This validates the overall global search methodology. More specifically, the feasible finite burn conversion validates that our estimates for representative finite burn maneuvers sufficiently constrain the impulsive solution space.

5.4 Summary

This chapter applied the search methodology developed in Chapter 3 to the GTOC4 problem, and found impulsive solutions exceeding the best known. The method uses no a priori information of these best known solutions, but rather adaptively updates parameters of the search to find increasingly better solutions over time. We var-

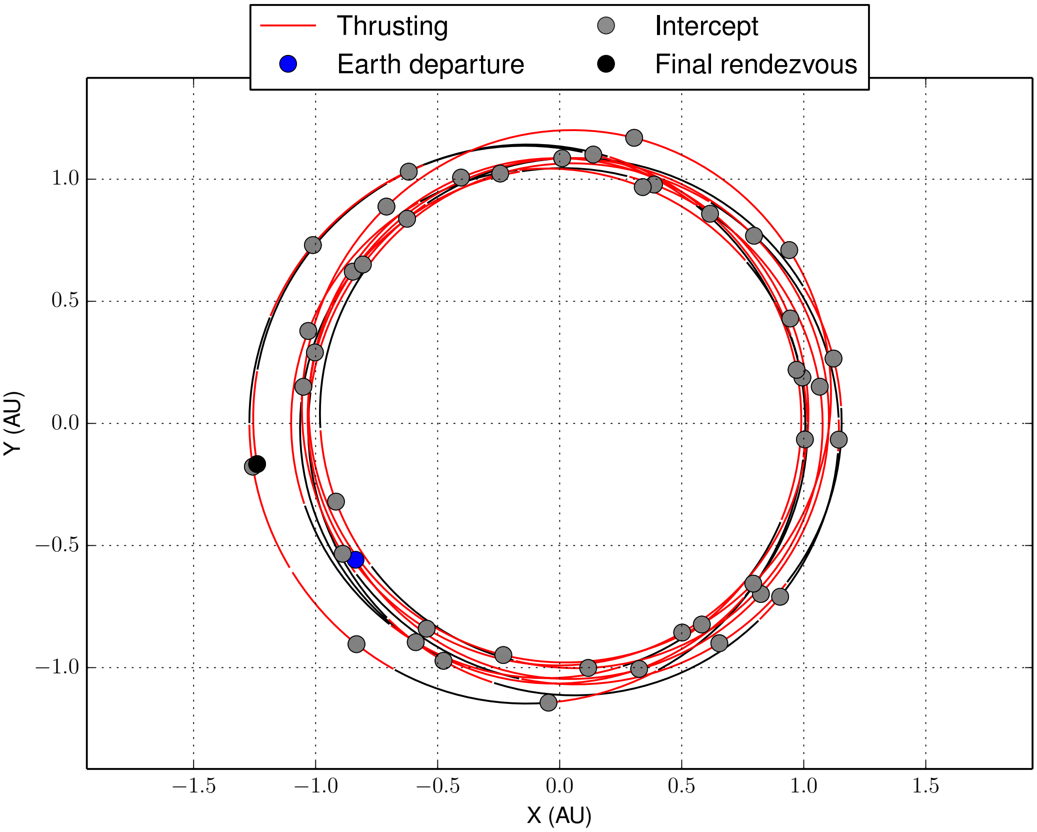


Figure 5.25: Low-thrust finite burn trajectory corresponding to the $J = -43$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.18 and Table 5.13.

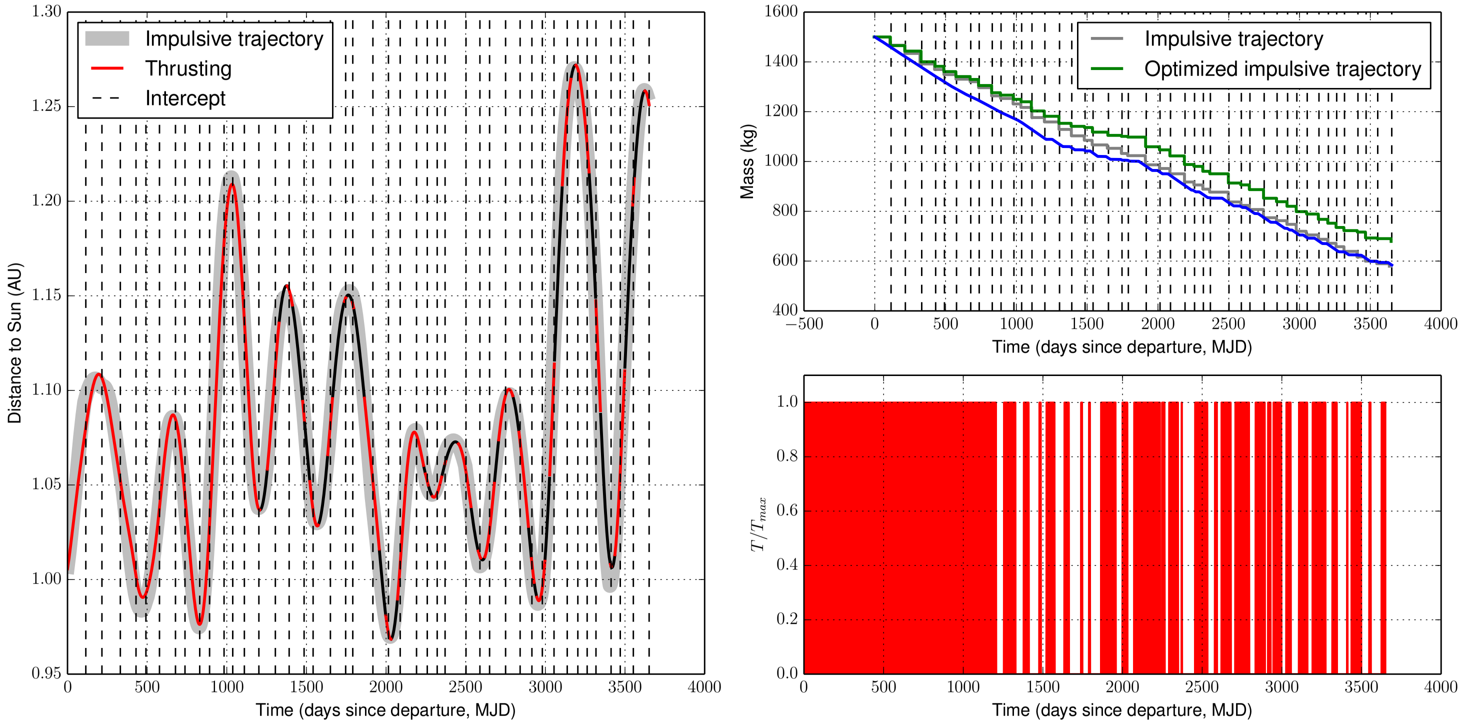


Figure 5.26: Low-thrust finite burn trajectory corresponding to the $J = -43$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.18 and Table 5.13. The original impulsive trajectory is also shown.

no.	t (MJD)	mass (kg)	Asteroid	no.	t (MJD)	mass (kg)	Asteroid
0	58597.39	1500.00	Earth	1	58712.38	1457.94	2000QW7
2	58813.53	1420.30	2003YT70	3	58929.01	1377.53	2005EZ169
4	59027.42	1341.40	2007LF	5	59089.62	1319.57	175729
6	59174.83	1290.98	2004UT1	7	59275.26	1260.41	2004XG
8	59334.95	1244.26	2005VO	9	59427.34	1215.52	2004MS1
10	59489.27	1197.00	2005WK4	11	59580.26	1174.04	1999VM11
12	59634.77	1156.66	2004BF11	13	59707.72	1127.75	152563
14	59798.87	1091.59	2007DS84	15	59902.33	1068.51	2003OQ13
16	59988.93	1054.56	2003WY153	17	60082.70	1043.16	2008JP24
18	60139.25	1033.89	2008CP	19	60247.89	1013.21	2008WM
20	60343.62	1005.01	2008CY118	21	60388.42	1003.53	2001TB
22	60514.72	980.09	2007YM	23	60612.26	956.87	2001SE270
24	60686.02	943.55	2005YP180	25	60788.65	902.69	2006SD25
26	60855.97	880.71	2006WG130	27	60918.35	866.50	2006BY8
28	60967.84	854.57	2002VX91	29	61099.21	833.68	2004FD
30	61186.52	817.66	2008TC3	31	61247.22	803.61	2005GC120
32	61347.72	773.80	2007CC27	33	61438.96	752.68	7335
34	61513.01	730.21	2008CN116	35	61578.52	711.02	2004RQ252
36	61651.78	693.46	2006BZ147	37	61734.71	679.68	1995CR
38	61801.32	665.82	2008EE9	39	61859.87	642.32	2007DB83
40	61915.18	636.55	2008UB7	41	62009.96	623.66	2003SW130
42	62067.65	610.20	2004JN1	43	62149.77	597.06	2005YK
44	62248.87	583.91	2006UB17				

Table 5.16: Low-thrust finite burn trajectory corresponding to the $J = -43$ impulsive trajectory of the reduced spacecraft performance case shown in Figure 5.18 and Table 5.13. The tour itinerary is shown.

ied several components of the search to study their impact on the results. We found that the finite burn feasibility constraints reduce the feasible solution space by orders of magnitude. Additionally, we found the search space pruning approach developed in Chapter 4 accelerates the search by an order of magnitude. The dynamic neighborhood selection procedure provides more diverse solutions through its dynamic intensification and diversification of the search. Finally, we generated optimal low-thrust finite burn trajectories for selected impulsive trajectories, and found a solution exceeding the best known for GTOC4.

The search procedure developed in this work is meant as the first stage of a multi-stage workflow. It finds impulsive solutions that are constrained in a way that helps ensure they can be converted to low-thrust finite burn trajectories. However, this conversion is not guaranteed to be feasible. In the future a second stage of this workflow should be developed that automatically converts and optimizes selected impulsive solutions into low-thrust finite burn trajectories.

Chapter 6

Conclusions

6.1 Dissertation Summary

This dissertation describes the spacecraft tour trajectory optimization problem, and develops an overall methodology for finding promising solutions. The method is based on many of the tenets of tabu search, and represents an automated, adaptive and efficient approach to finding globally good solutions to a broad class of spacecraft tour trajectory problems.

Chapter 2 begins by developing a general model and parameterization for tour trajectories. This model is a complication to the traveling salesman problem, where now both the agent and targets objects are allowed to move with time according to a set of prescribed dynamics. A model for spacecraft tour trajectories subject to two body dynamics and impulsive maneuvers is then presented. Constraints are developed on impulsive maneuvers to prune away solutions that are unlikely to have equivalent finite burn trajectories. These additional constraints are only approximations, but allow the model to be used for preliminary design for low-thrust trajectory optimization problems as well. This augmented model is used in later applications to the fourth annual Global Trajectory Optimization Competition problem (GTOC4).

Chapter 3 develops the overall global search methodology, and is the core chapter of the dissertation. It begins by implementing the building blocks common to local search algorithms: the solution representation, neighborhoods, and objectives. A tree-based solution representation is defined that is shown to be especially efficient for problems with expensive solution evaluations. Neighborhoods are then defined to operate on this representation, leading to a neighborhood definition that allows for strategic intensification and diversification during the search. A guiding objective is then developed which allows the algorithm to adaptively target increasingly better solutions as the search progresses. Finally, these components are combined within the context of the tabu search algorithm. Tabu attributes and recency-based tabu memory are presented, as well as a strategic intensification and diversification approach that dynamically adjusts the neighborhood to ensure a sufficiently diverse exploration of the search space. The final result of the chapter is a tabu search algorithm applicable to general tour trajectory optimization problems. The algorithm is deterministic and generates a diverse population of feasible solutions.

Chapter 4 develops a numerical search space pruning procedure which can be used to improve the performance of the tabu search algorithm. The chapter first describes the brute force approach, and then develops the concept of a trajectory envelope that bounds the reachable domain of the spacecraft. A simple numerical procedure is then developed to compute an upper bound on the trajectory envelope. The procedure is based on the generation of bounding boxes in space and time. Target objects intersecting these bounding boxes may be feasibly reached by the spacecraft; those not intersecting can be safely pruned from the search space. The pruning condition is thus a necessary, but not sufficient, condition for reachability. The performance of the space pruning method is analyzed for a simple example according to an assumed discretization, and a significant reduction in the number of required Lambert targeting computations is shown, which should yield speedups in real applications.

Chapter 5 then applies all of these components together to the fourth annual Global Trajectory Optimization Competition (GTOC4). The impulsive tour model is extended with constraints specific to GTOC4, and the tabu search algorithm is executed for a broad set of cases. Each case is executed in parallel on the Stampede supercomputer to generate a distribution of results across a range of launch epochs. The algorithm finds impulsive solutions meeting or exceeding every solution from the original GTOC4 results. Impulsive solutions are also found exceeding the currently best known GTOC4 solution. A sensitivity analysis is then conducted, examining the impact of components of the search algorithm including the finite burn feasibility constraints, search space pruning procedure and dynamic neighborhood selection procedure. In particular, the search space pruning approach is found to yield an order of magnitude speedup in performance. We then examine cases where the spacecraft is assumed to have reduced performance through lower thrust capability and available fuel to see the impact of solution quality. Finally, we convert solutions from the reduced performance cases into fully optimized low-thrust finite burn solutions, making use of an indirect optimal control approach provided by Olympio. The results are new solutions to the GTOC4 problem, including one exceeding the best known solution in the literature.

6.2 General Conclusions

The primary contribution of this work is the development of a tabu search methodology for the spacecraft tour trajectory optimization problem. This work represents the first application of tabu search to spacecraft trajectory optimization, and it is the hope of the author that such methods will find more practical applications in the future. The resulting algorithm represents an automated and adaptive approach for finding promising solutions to tour problems, and yields a broad population of feasible solutions in short run times. Another contribution is the extension of tabu search to tree-based solution representations and neighborhoods. The tree-based representation leads to increased speed and efficiency on problems with expensive solution evaluations. We believe this is the first time a tree-based solution representation has been used with a tabu search algorithm. The search space pruning procedure based on trajectory envelopes is also novel, and could be applicable other problems with different trajectory dynamics. Finally, the work resulted in a collection of new solutions to the GTOC4 problem, including a new solution exceeding the previously best known.

6.3 Future Work

This work does not represent an exhaustive exploration of search approaches for spacecraft tour trajectory optimization, nor does it fully explore the ways in which tabu search can be used to attack such problems. It is the hope of the author that future work is able to build upon and improve the methods developed here. Although the tabu search approach has been demonstrated successfully on the GTOC4 problem, it should also be applied to other problems to further test its applicability and performance. The ability to adapt the definitions of the solution representation and neighborhoods should make the algorithm easily applicable to diverse problem types, in particular other GTOC problems. Further, applying it to the design of active debris removal trajectories addressing the Earth orbital debris problem could be fruitful.

A particular area for future work is the parallelization of the algorithm. Although we ran the software on large-scale supercomputing clusters, we did so by running independent instances for varying launch epochs. Another approach is to parallelize the solution representation and neighborhood computations directly. The tree-based solution representation is especially amenable to distributed parallelization, as subtrees of the overall search tree can be distributed across a supercomputing cluster and computed on in an embarrassingly parallel manner. Such a parallelization would further improve the speed of search and possibly aid in the discovery of improved solutions to challenging problems in spacecraft tour trajectory optimization.

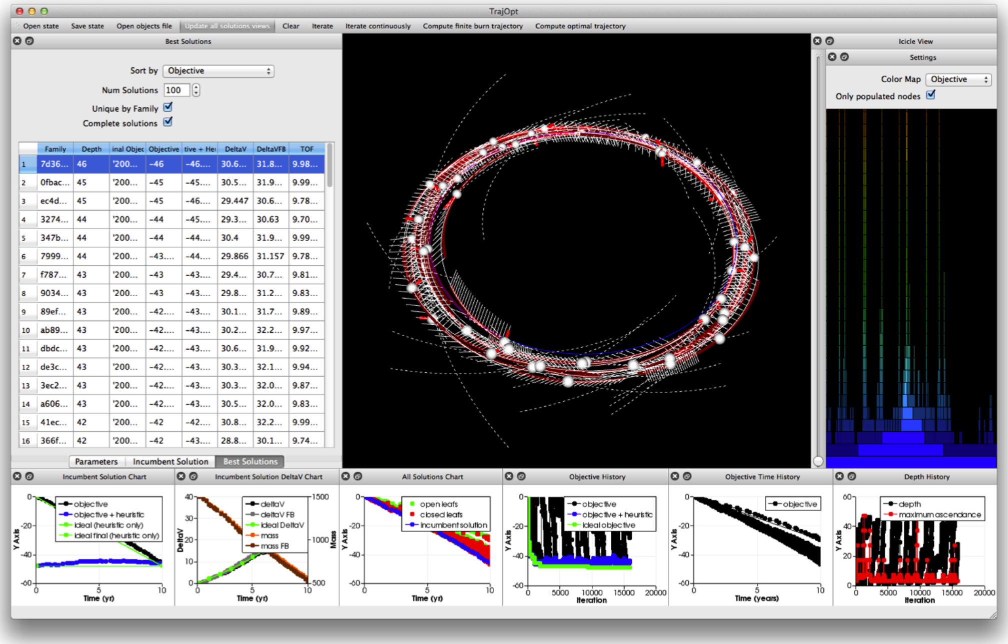
Appendix A

Software Implementation

A software package was developed that implements all components of the tabu search methodology presented in this work. This cross-platform application has a graphical user interface and visualization capabilities that allow users to interactively run, adjust parameters and analyze results of the tabu search algorithm (Figure A.1). Additionally, the application can run in batch mode for large-scale studies, such as those executed in Chapter 5 on the Stampede supercomputer at the Texas Advanced Computing Center [2]. The application, written in C++, relies heavily on an object-oriented programming model. This section briefly describes the implementation of the tree-based solution representation used throughout the work as well as the neighborhoods built on that representation. The implementation is described using C++ terminology; however the strategy described here can be implemented in other object-oriented programming languages.

A.1 Tree-Based Solution Representation

The tree-based solution representation developed in Section 3.1 is implemented using an object-oriented approach. Recall that branches of the tree represent trajectory



Sort by: Objective

Num Solutions: 100

Unique by Family ☑

Complete solutions ☑

	Family	Depth	inal Objec	Objective	tive + He	DeltaV	DeltaVFB	TOF
1	7d36...	46	'200...	-46	-46....	30.6...	31.8...	9.98...
2	0fbac...	45	'200...	-45	-45....	30.5...	31.9...	9.99...
3	ec4d...	45	'200...	-45	-46....	29.447	30.6...	9.78...
4	3274...	44	'200...	-44	-45....	29.3...	30.63	9.70...
5	347b...	44	'200...	-44	-44....	30.4	31.9...	9.99...
6	7999...	44	'200...	-43....	-44....	29.866	31.157	9.78...
7	f787...	43	'200...	-43	-43....	29.4...	30.7...	9.81...
8	9034...	43	'200...	-43	-43....	29.8...	31.2...	9.83...
9	89ef...	43	'200...	-42....	-43....	30.1...	31.7...	9.89...
10	ab89...	43	'200...	-42....	-43....	30.2...	32.2...	9.97...
11	dbdc...	43	'200...	-42....	-43....	30.6...	31.9...	9.92...
12	de3c...	43	'200...	-42....	-43....	30.3...	32.1...	9.94...
13	3ec2...	43	'200...	-42....	-43....	30.3...	32.0...	9.87...
14	a606...	43	'200...	-42....	-43....	30.5...	32.0...	9.83...
15	41ec...	42	'200...	-42	-42....	30.8...	32.2...	9.99...
16	366f...	42	'200...	-42	-43....	28.8...	30.1...	9.74...

Color Map: Objective

Only populated nodes ☑

Figure A.1: The software package that implements the tabu search methodology allows users to interactively run, adjust and analyze results of the search through its user interface and visualization capabilities.

Method	**Description**
getParent()	returns parent node of the current node
getChildren()	returns children node(s) of the current node
getAncestor(h)	returns h^{th} ancestor of the current node
getLeaves()	returns all descendant leaf node(s) of the current node

Table A.1: Tree traversal methods implemented in the *TreeNode* class.

segments and their decision variables, and nodes of the tree correspond to states at the segment boundaries. Each node of the tree may have multiple children nodes, and except for the root node, each has a single parent node. Results of state, objective and constraint evaluations are stored in the tree, allowing for efficient generation of new solutions through iterative expansion of the tree.

The tree and all of its basic traversal operations are implemented using a single class in C++ named *TreeNode*. This class contains member variables representing the trajectory segment decision variables and evaluations as well as member methods (or functions) that operate on this data. The class also contains pointers that reference children nodes as well as the parent node. When this class is instantiated it forms an object. Any number of these objects can be instantiated, and through assigning the children and parent pointers, the tree structure is formed. Figure A.2 shows a *TreeNode* object and both the parent and children nodes it references.

The methods defined in the *TreeNode* class allow for traversing the search tree through use of the parent and child node pointers. Table A.1 lists several of these methods. The getParent() and getChildren() methods are simple: they simply return the member pointers to the corresponding *TreeNode* objects. The remaining methods, getAncestor(h) and getLeaves(), rely on recursion to compute their results. Algorithm 13 gives pseudocode for a getLeaves() imple-

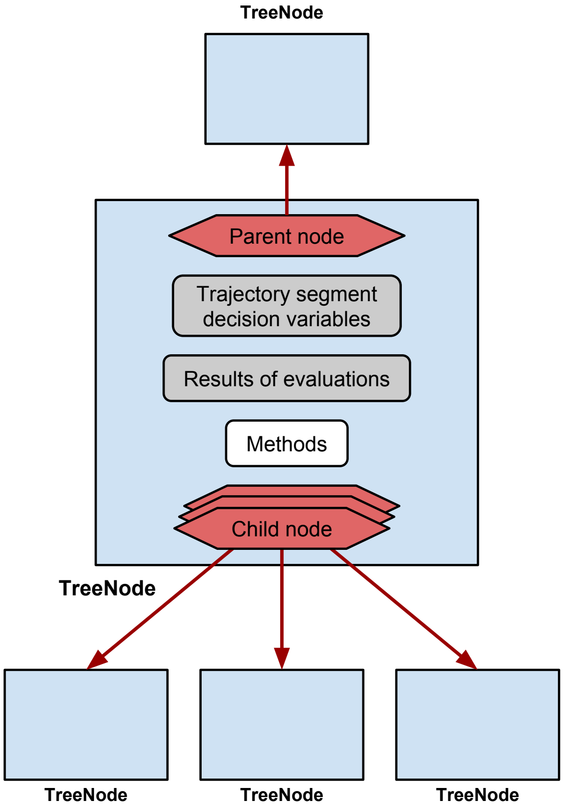


Figure A.2: The *TreeNode* class represents nodes in the tree-based solution representation. A *TreeNode* object is shown with the parent and children nodes it references. The collection of *TreeNode* objects combine to form the search tree.

mentation. For a large tree, the GETLEAVES() method will be called recursively many times to populate the full list of leaves. Other operations on the tree-based solution representation are implemented in a similar manner. For example, when a *TreeNode* object is deleted from memory, upon destruction it will recursively delete all of its children nodes and their descendants. In this manner, entire subtrees of the overall search tree can be easily pruned.

Algorithm 13 Example GETLEAVES() implementation.

```
▷ return all descendant leaf node(s) of the current node
▷ on initial call, leafNodes is an empty list
▷ upon termination, leafNodes contains the full set of leaf node(s)
procedure GETLEAVES(leafNodes)
    ▷ get list of children of current node
    children = this → GETCHILDREN()
    ▷ if there are no children, add this node to the list of leaf nodes and return
    ▷ otherwise, continue recursively to find the remaining leaf nodes
    if LENGTH(children) == 0 then
        leafNodes = leafNodes + this
        return
    else
        for child in children do
            child → GETLEAVES(leafNodes)
        end for
    end if
    return
end procedure
```

Neighborhoods can also be computed using these basic tree traversal methods. Recall the restricted best-first neighborhood presented in Section 3.2.2, defined as

$$\mathcal{N}(\mathbf{x}, h) = \mathcal{C}(\mathbf{x}) + \mathcal{L}(\mathcal{P}^h(\mathbf{x})) \tag{A.1}$$

In this case the solution $\mathbf{x}$ is a node of the tree. This neighborhood is composed of the children of the current node $\mathbf{x}$ (which are generated on-demand) and the leaf nodes of the h^{th} ancestor of $\mathbf{x}$. The two terms of the neighborhood can then concisely be computed as

$$\mathbf{neighborhood} = \mathbf{x} \rightarrow \textsc{generateChildren}() \tag{A.2}$$

$$\mathbf{x} \rightarrow \textsc{getAncestor}(h) \rightarrow \textsc{getLeaves}(\mathbf{neighborhood}) \tag{A.3}$$

We limit the discussion here to the implementation of the basic tree data structure and operations required by the search. The object-oriented approach gave a great deal of flexibility in the search algorithm design, and simplified the overall implementation of the tabu search methodology.

Appendix B

Set of GTOC4 Asteroids

Name	**a** (AU)	**e**	**i** (deg)	**LAN** (deg)	**arg periapsis** (deg)	**M** (deg)	**Epoch** (MJD)
’Earth’	0.99998805	0.016716812	0.000885435	175.406477	287.6157755	257.6068371	54000
’1580’	2.196803375	0.487683107	52.0907939	62.32479512	159.5398386	3.521686818	54800
’1620’	1.245550856	0.335510678	13.33769341	337.2660168	276.8065388	136.950031	54800
’1943’	1.430317185	0.25581586	8.704064516	246.4019152	338.2538753	288.4052376	54800
’2061’	2.264953422	0.537119092	3.770822489	207.6541081	156.4320284	46.9954969	54800
’2135’	1.599571803	0.503267841	23.05431224	191.2628488	290.8388917	256.7249811	54800
’2201’	2.172221847	0.71280397	2.51676523	76.61936199	96.22575966	353.1160627	54800
’2329’	2.404555168	0.657495712	24.42995952	169.4393808	145.8319771	235.8255564	54800
’2340’	0.844210764	0.449758341	5.854788239	211.5046016	39.99419575	240.4482744	54800
’2368’	2.104871082	0.413837066	5.237068616	287.5929827	42.60536676	87.28448582	54800
’3199’	1.57448481	0.283803324	32.96957152	340.0439748	53.37489659	65.45870912	54800
’3352’	1.878651159	0.369609614	4.774066184	107.4215245	15.91244872	288.4003058	54800
’3361’	1.209433393	0.322874052	2.684893669	189.6018219	301.6452091	49.38366372	54800
’3551’	2.093250599	0.4866558	9.503739583	173.8729195	193.1758024	114.8907422	54800
’3671’	2.198421931	0.541925061	13.54629618	82.19859996	204.2248653	177.8625945	54800
’3752’	1.413609555	0.301934792	55.55533428	147.9932027	312.2097902	207.086479	54800
’3838’	1.504735141	0.702152155	29.24191496	235.6247583	49.57599503	9.295677694	54800
’4015’	2.637358396	0.624226876	2.785634942	270.5551279	91.25972407	284.9655895	54800
’4179’	2.531891004	0.628756787	0.446055628	124.2914647	278.7504608	5.856710222	54800
’4183’	1.981401438	0.63605604	6.750976852	295.6387792	235.4235865	288.9865162	54800
’4197’	2.301850396	0.77243627	12.27416923	9.094393488	120.3567665	155.1451023	54800
’4401’	2.579475249	0.565697433	26.64396129	23.14640005	68.00072541	194.7461434	54800
’4581’	1.022340642	0.356990879	4.912672961	180.3796973	255.2124382	82.96810044	54800
’4769’	1.063106675	0.483276505	8.888351558	325.641394	121.3283773	149.8263685	54800
’4953’	1.620966177	0.657184694	24.40880111	77.92573895	77.54990252	35.68316165	54800
’4954’	2.001241603	0.448627643	17.44936075	358.560784	52.42077466	129.1405337	54800
’5324’	2.961804862	0.614749968	19.4955921	353.0854483	320.1433509	69.50895951	54800
’5381’	0.947472265	0.296151529	48.96972507	58.55949159	37.42691489	120.5688032	54800
’5496’	2.433767578	0.637130122	68.02254894	101.063151	118.13515	130.4864891	54800
’5836’	2.444042815	0.53307381	7.982279679	239.7403886	76.70188788	2.73534173	54800
’5879’	1.624537687	0.289357225	21.57451779	145.9020014	355.5520251	42.755942	54800

’6047’	1.454334288	0.352136234	23.4706	6.157745352	103.7241373	223.265127	54800
’6050’	2.203564288	0.435671439	6.401849655	88.43165461	284.6361028	84.7744825	54800
’6178’	2.810797107	0.585699102	4.308460392	64.80047655	127.1709546	286.5684651	54800
’6491’	2.502080004	0.589355142	5.736291828	304.0793167	320.6442509	140.9882493	54800
’6611’	1.695486073	0.485142564	8.691105208	231.0855529	281.1748921	247.1230966	54800
’7088’	1.980593158	0.390968965	8.302423916	102.7213092	354.7150648	21.33957576	54800
’7236’	2.726675994	0.558938898	16.32393379	308.5686477	337.967811	275.5454243	54800
’7335’	1.770682123	0.484335415	15.21284399	61.45731733	232.0635564	83.4831008	54800
’7822’	1.122713269	0.164604295	37.12224582	156.8869326	249.4491272	57.43900588	54800
’7888’	2.434533927	0.664254211	26.07679261	165.9632278	323.0514121	319.7774804	54800
’7889’	1.261607118	0.346373832	36.90786965	111.3076654	349.1251113	262.5499401	54800
’7977’	2.226427348	0.465318559	25.17583433	134.2720121	248.0245805	132.2103866	54800
’8037’	1.986173128	0.417996714	5.909083326	22.50606756	105.6917101	249.2985719	54800
’8566’	1.506610616	0.430722445	37.96443823	164.184782	125.1150488	238.1682175	54800
’10165’	1.234614611	0.503836635	23.89632955	312.4810833	348.389821	244.8915151	54800
’11284’	1.740183865	0.337880053	1.994816337	311.8533323	170.8187529	74.51135363	54800
’15745’	1.719608036	0.255020728	14.42364283	132.6924575	140.4725298	269.4994476	54800
’15817’	1.32470088	0.118163447	13.87174661	162.5436592	94.27673054	190.437439	54800
’18736’	2.355153205	0.487381265	2.857904445	298.754223	220.8942834	57.54143321	54800
’20425’	1.564747005	0.476410788	6.977300036	227.5209849	295.9391302	352.1284795	54800
’20429’	1.555971437	0.464074166	6.297251978	61.82626684	147.4952282	337.5648847	54800
’22753’	1.218706829	0.569861831	3.205236104	307.5937648	324.5706645	237.3954333	54800
’24443’	2.310334597	0.821838061	25.81427434	178.3058762	231.017156	119.5056066	54800
’25330’	1.540386481	0.370977534	14.32876201	50.6890964	85.85195384	19.44766289	54800
’26166’	3.300186332	0.644688186	14.79281632	185.7727227	62.87749458	97.96360211	54800
’26817’	2.809844016	0.592016883	3.486512037	153.282956	156.521087	199.2725031	54800
’29075’	1.698749638	0.507531452	12.18197322	356.7825853	224.5335567	161.059429	54800
’31346’	2.029336634	0.430606905	5.966392429	299.8201609	350.2553987	211.4706406	54800
’40263’	1.494876494	0.161115872	25.84356888	172.8663764	198.6014335	264.146218	54800
’52340’	2.209848082	0.549929889	8.045830458	5.867502095	115.4703029	280.1327446	54800
’52387’	1.282142847	0.189705143	24.15552334	297.6328423	195.4394356	6.895723957	54800
’52750’	1.427127526	0.525219207	11.16150198	141.4095756	334.0095306	113.8409431	54800
’53409’	2.100983516	0.62860865	10.80502183	207.0195274	147.2200228	8.434473167	54800
’54660’	1.476664703	0.280924976	46.68459728	223.7485473	157.9271945	182.2056258	54800
’54686’	1.776491378	0.341909194	33.20560209	161.7916281	265.8687923	153.5615067	54800
’55532’	1.794762081	0.695961054	38.47314311	81.58369333	132.2883251	276.1954517	54800
’65733’	1.154166638	0.474510895	4.15615228	337.5049656	168.1712913	303.178422	54800
’67399’	1.301190105	0.346028973	14.69838053	332.9459236	225.1701241	314.7159721	54800
’68267’	1.50942487	0.42770736	38.81869472	8.065275279	317.3510195	321.2861488	54800
’68346’	1.507367363	0.416676208	16.68730349	219.467612	140.142533	285.2511937	54800
’68348’	2.15229294	0.842163684	25.44436903	236.2814352	181.5422624	43.61374569	54800
’68372’	1.618224127	0.415657856	8.09481772	253.1304042	322.0808376	260.974575	54800
’85770’	0.998587664	0.344931005	33.17892204	18.39094108	234.3488472	144.8679418	54800
’85774’	1.404102635	0.329549296	13.59119125	64.72582717	49.99786202	339.8609108	54800
’85804’	1.721249491	0.354226419	27.66244479	285.8425937	269.7262061	59.62985604	54800
’85818’	1.656659573	0.416943184	62.70177829	235.6841505	301.3033528	145.4152686	54800
’85867’	1.83044366	0.301975239	0.942737744	254.7382609	286.9289404	315.4889281	54800
’85938’	1.852525771	0.483076591	9.149753901	20.00323414	197.5512351	295.9403189	54800
’86666’	1.462938703	0.426851765	29.01462618	187.0054919	258.8274099	10.71355278	54800
’86667’	0.859290751	0.59466828	14.28367694	208.394114	172.4010033	118.2779807	54800
’86878’	1.341664031	0.618518983	9.474205434	231.1173124	214.7637777	235.7586408	54800
’87025’	1.226752294	0.48369449	25.32151624	120.5474333	359.5390192	158.519311	54800
’88188’	2.006621285	0.392973328	11.39165586	340.3315041	194.8070711	234.2302096	54800

’88213’	0.953955927	0.595319797	17.81513774	114.3112178	194.9494557	299.0335363	54800
’88254’	1.181885041	0.629434958	1.524048634	272.5891799	139.7134338	39.25672262	54800
’88263’	2.096851272	0.431466503	38.81746237	232.9576842	241.4817739	236.6536192	54800
’89355’	1.78698398	0.307979262	22.67116814	103.2034548	84.77086051	291.7632555	54800
’89958’	1.642025674	0.88616959	9.963631901	188.5371693	222.5275178	348.6287744	54800
’90147’	1.474132433	0.331548616	27.99181896	282.7752048	104.9911597	127.1275253	54800
’90373’	1.626719395	0.204626083	9.871523725	189.483247	218.605361	161.870993	54800
’96744’	2.088694973	0.780130777	35.23004106	196.4956261	35.3275688	97.10859278	54800
’99907’	0.728523907	0.594662556	28.79484776	225.6204498	2.818189851	46.50983881	54800
’101869’	1.623770941	0.610707599	4.764751493	111.1241734	268.5582825	171.0534702	54800
’108519’	1.60334332	0.270758773	16.39241798	267.349534	343.7148912	260.212813	54800
’115052’	1.668771748	0.318153219	31.31981816	190.5983676	88.16274908	206.6199823	54800
’136617’	1.637015962	0.416973711	4.634071975	268.7839676	24.74307724	256.2013024	54800
’136745’	2.365179413	0.483637689	17.77380099	248.0911341	131.5166154	221.7052074	54800
’136793’	1.146997931	0.46532235	17.38035169	296.3220823	36.95020666	339.0497247	54800
’136795’	1.745578148	0.478466325	11.00230925	50.28077213	147.335731	23.03013041	54800
’136818’	0.937516093	0.346476757	12.7723709	260.0393953	203.7323821	18.36475196	54800
’136849’	1.492708101	0.57835311	7.798230669	110.9539571	97.40055875	306.3685345	54800
’136923’	2.134151268	0.442260879	6.61900528	51.32713015	286.9037093	50.99542533	54800
’137064’	1.374171498	0.19529683	19.50169977	36.18398842	97.49879028	30.05835516	54800
’137078’	1.939006551	0.639190224	23.20768442	324.5398769	9.461415035	284.8944892	54800
’137126’	1.772582569	0.599487824	5.545796776	157.3554951	89.89392987	19.44823082	54800
’137170’	0.819000805	0.462408512	25.66262523	155.9171598	253.3569392	98.03128354	54800
’137199’	1.457304441	0.29272176	16.57247087	105.00906	76.28470007	202.3482294	54800
’137802’	1.77661438	0.35183578	31.584273	116.5086618	272.5529736	319.6560879	54800
’137805’	0.829417131	0.558351642	16.74206645	349.6469436	292.7593644	81.76134505	54800
’138175’	1.004744986	0.293434217	5.241567706	25.94844279	280.9125953	133.7812764	54800
’138205’	2.572119285	0.619018839	11.04862802	5.217152095	304.0430921	355.2085959	54800
’138325’	2.164510393	0.804026781	25.5863925	173.4596962	164.2696747	225.3774374	54800
’138359’	1.14127737	0.361304818	20.24014291	44.04001146	4.676807639	177.6810939	54800
’138524’	2.362045766	0.565326178	6.213428735	226.7129144	181.0904829	70.27235765	54800
’138847’	1.618789112	0.287526374	22.17896106	207.1708585	16.15509619	150.3717449	54800
’138859’	1.573461425	0.531641523	13.1427338	56.02835702	144.2434605	289.0824843	54800
’138893’	1.172912905	0.743622469	18.33146209	265.3273	341.2849035	6.746425751	54800
’138911’	1.349690546	0.081610559	1.660676415	171.511561	42.9268535	273.7541041	54800
’138971’	1.034781446	0.333641021	7.903356226	353.8533776	271.6844622	82.28247079	54800
’139289’	1.2595377	0.841214664	23.21835873	102.9948392	291.2181077	64.29201543	54800
’140158’	1.347082459	0.460823212	2.513012139	126.9377669	42.55450599	38.69557353	54800
’140928’	1.518728493	0.296909829	20.66254158	245.547743	257.218896	194.6091057	54800
’141018’	1.398528121	0.24104933	2.866921953	91.85186236	100.9683262	351.0646663	54800
’141078’	1.862935783	0.450973288	11.46784377	234.8885659	270.5064089	225.2037989	54800
’141424’	0.979708095	0.176707303	6.878981113	8.737677973	331.5935572	144.2108945	54800
’141484’	0.857597351	0.369433062	16.60285192	234.3266209	94.07338159	284.8363857	54800
’141527’	1.514018807	0.626759774	9.195853842	187.7139153	247.427582	253.2030183	54800
’141593’	2.001531876	0.530120513	2.360340563	307.1780805	2.02942708	82.9188566	54800
’141851’	1.197996788	0.850017829	19.22098967	304.2933442	224.2376902	331.4690526	54800
’142348’	2.065864969	0.458107007	6.06203918	96.75231763	324.1028353	4.129807776	54800
’142464’	1.233365066	0.154348907	16.27789665	191.8926309	29.21085413	332.1986528	54800
’143409’	1.949386122	0.351037924	8.165153317	163.9115661	44.1164357	9.454139923	54800
’143527’	1.661484437	0.351925358	17.3106945	181.8024765	242.0822366	319.8244601	54800
’143947’	2.180557165	0.655158898	21.00430673	217.6446355	135.5921114	251.1026794	54800
’144861’	2.511605134	0.747854978	39.40286721	159.251881	199.4036456	349.4737141	54800
’145656’	2.62939026	0.561173352	11.02900344	176.8144361	97.95384724	118.0720336	54800

’152561’	1.454612041	0.485515401	19.58482121	359.4199756	68.8353691	260.9574389	54800
’152563’	0.908038661	0.271823498	7.254113017	315.4681181	336.4302657	17.5656488	54800
’152575’	2.685129552	0.524396541	12.33108803	33.80387635	190.5615433	114.3651481	54800
’152742’	0.878266762	0.739126431	13.43603598	280.0958637	353.0725613	151.5625478	54800
’152895’	2.288361947	0.494232412	2.983254668	29.91508906	173.8693449	167.8069601	54800
’153002’	1.340611486	0.795645177	31.47615081	213.2194226	233.3050461	224.6951264	54800
’153195’	1.301297795	0.619045412	41.11364232	21.48247534	262.8741887	208.3488402	54800
’153249’	2.117879975	0.590894572	41.21392332	329.0210753	297.9237421	119.9303065	54800
’153267’	1.781585579	0.614162604	9.659820988	75.67633588	330.334646	124.1436647	54800
’153306’	2.552069358	0.52354033	26.98349642	226.7311328	272.5337589	335.4574798	54800
’153315’	1.237488472	0.4497061	34.6959012	112.6067591	288.3551273	55.55647625	54800
’153460’	1.413822196	0.581028137	10.09321265	211.5781371	30.25052966	146.7139543	54800
’153792’	2.104925334	0.739112797	10.69119804	277.3734862	243.0279652	77.00195325	54800
’154144’	1.834033296	0.296257438	23.61083392	172.6662506	126.2440294	172.6756262	54800
’154244’	2.308247656	0.548886131	3.236898587	214.9045038	95.98028344	293.9412337	54800
’154276’	1.705709894	0.68990451	8.744831299	34.40099908	99.31485112	240.0697741	54800
’154347’	1.849917384	0.691913883	17.81432896	331.8105514	24.71506523	162.7846473	54800
’154991’	1.704954244	0.322952841	5.637623681	245.700687	269.2153475	289.3650595	54800
’155110’	1.261393147	0.348316939	30.38683148	226.2333639	44.71994178	125.3562108	54800
’159368’	2.331231324	0.442583998	3.354671556	342.412487	12.16223919	69.66271563	54800
’159399’	1.526966571	0.21407673	41.96430846	214.9000642	353.1036303	299.4964988	54800
’159459’	2.340591966	0.79650335	56.02443317	185.4523728	185.0552943	119.90976	54800
’159504’	2.432958001	0.618287761	9.70283995	107.8070033	237.6186999	72.49907729	54800
’159533’	1.654667608	0.288928749	12.85126992	71.71691828	275.7979811	109.3381647	54800
’159555’	1.630387552	0.230091585	29.43376646	215.0079911	123.417784	179.108276	54800
’161995’	2.287905139	0.47818729	25.26696316	81.27775681	220.7054199	115.923422	54800
’162039’	1.802188278	0.660369157	5.280127202	53.09786244	279.945133	42.98338218	54800
’162080’	0.896685587	0.358217996	16.20857228	344.4124973	356.8079102	71.32535153	54800
’162082’	1.246143989	0.187064235	20.04528294	213.6065487	148.4523757	79.41694601	54800
’162157’	1.297150145	0.351560708	15.26385593	132.050521	279.2290826	283.0036694	54800
’162173’	1.189708611	0.190336053	5.88339684	251.6413476	211.4098741	243.1297112	54800
’162196’	1.826382443	0.374934128	22.44043354	172.0391229	234.1427255	231.4929225	54800
’162210’	2.295139395	0.696321825	5.20986728	327.7911059	319.1032767	243.0870171	54800
’162215’	1.081404466	0.436518421	17.33462851	202.3886911	346.7011769	238.5222534	54800
’162273’	1.59365182	0.236077793	20.18417428	234.4670267	40.87056367	302.4392615	54800
’162416’	1.853851964	0.477534444	0.393857923	215.3540621	18.92938397	139.7685629	54800
’162470’	1.112691911	0.552821121	35.28404293	83.79906314	31.73425416	152.4796746	54800
’162566’	2.636859035	0.573468502	13.84740096	331.4001654	143.149663	284.3468138	54800
’162581’	1.667645759	0.351554299	15.89063036	200.4320363	248.050929	244.4466153	54800
’162873’	1.400631967	0.086489309	20.19672217	357.9468463	198.5972131	220.0305422	54800
’162913’	1.271098777	0.51960266	8.639820111	170.6166023	356.0451564	133.3721046	54800
’162922’	1.318085021	0.381629656	10.29819223	284.3243426	291.1059919	357.3656081	54800
’162979’	2.035944399	0.545406523	17.08398093	311.6721368	325.9695738	192.67716	54800
’162980’	1.553094714	0.488898441	30.39075163	177.673755	351.3538362	93.37027543	54800
’163191’	1.836355093	0.464054753	16.30285242	179.250576	44.10591786	232.3880181	54800
’163250’	2.692448889	0.538249111	34.97028805	170.3155238	350.9397579	185.552618	54800
’163252’	2.13080607	0.439920633	9.000471127	95.70720719	200.8596804	12.83836698	54800
’163295’	2.474582565	0.64005412	5.822522981	33.12795494	76.72169162	267.0606029	54800
’163335’	1.327976502	0.667216346	56.28635426	247.0797416	155.6111193	33.43179437	54800
’163364’	1.364276095	0.368725797	4.17500486	260.1622684	274.9125884	100.0498607	54800
’163732’	2.752731464	0.695906155	44.61953735	193.4239356	190.6045351	44.92608391	54800
’164121’	1.10997633	0.291947144	44.05984819	38.3624784	90.94344263	25.45138439	54800
’164202’	0.989436124	0.279691619	4.66341354	343.4112319	55.76853507	23.60924836	54800

’164207’	1.000827922	0.136404427	13.64799725	38.83385442	280.9710469	115.7980815	54800
’164215’	2.114956167	0.396922416	4.879021978	236.3292023	76.13584346	147.8671941	54800
’164216’	2.153500086	0.563962646	19.89511929	295.2837167	326.1704499	156.6873993	54800
’164217’	2.013107188	0.415931563	48.89984284	145.0244786	178.9283016	180.900169	54800
’164294’	0.617714332	0.454270094	2.9536299	211.7945395	4.763826765	236.1438523	54800
’164341’	1.626795168	0.254143343	13.01421275	140.7341333	42.71066437	275.5504947	54800
’164400’	1.6565537	0.467601183	6.625671594	219.0499604	202.91902	8.574228375	54800
’170013’	2.963934399	0.801025288	24.04526153	185.9768161	328.278736	53.13334281	54800
’172722’	1.543646849	0.699450801	7.232599973	341.40854	17.04284038	213.7637546	54800
’173689’	1.779628035	0.394589426	10.40737755	273.1393456	313.6132043	89.75993231	54800
’174806’	2.515581128	0.572610648	11.35407493	318.990632	204.877171	52.35016765	54800
’175189’	2.056423411	0.3874061	2.62398849	169.6302451	78.4462168	34.54314366	54800
’175706’	1.054271178	0.349872214	1.990349109	299.8810867	23.93032639	150.918598	54800
’175729’	1.2720696	0.424807259	11.53689051	124.3797596	259.1173302	255.1302962	54800
’177016’	1.15995231	0.582116888	13.79999349	137.6573047	155.2632898	134.5702654	54800
’177651’	1.154377745	0.698853767	42.40955431	89.44742819	186.3192465	157.2931357	54800
’185851’	1.365429567	0.376716445	8.669995771	358.7831634	289.6529655	79.07800391	54800
’186823’	1.205114893	0.676862075	21.94828868	190.1243135	161.0586363	105.325275	54800
’186844’	2.436984235	0.669293502	7.792965521	261.9084756	54.22102069	49.15142779	54800
’189011’	1.500140847	0.232081945	18.6937839	254.496593	130.2552306	346.0843383	54800
’189173’	1.843861598	0.570310269	43.0717803	269.0523425	273.7397206	106.8198136	54800
’189263’	2.729666967	0.588355038	16.7446469	202.1491882	203.9260484	340.6959272	54800
’190119’	2.462719261	0.891004755	30.05670457	225.5732001	19.84805084	33.79449573	54800
’190161’	2.23752586	0.454335041	3.993268983	305.1122919	55.55790772	358.9380411	54800
’190208’	2.053666545	0.486408565	4.080519795	326.2190122	104.8024463	340.7616372	54800
’190758’	1.749197744	0.364558754	13.9446294	178.7043475	130.2620821	69.06699258	54800
’190788’	1.268260084	0.791939539	20.13557116	350.0674973	199.5246917	107.9812156	54800
’191094’	2.113591041	0.641885465	32.20036631	349.7461255	259.1067811	50.57046495	54800
’192563’	1.45214248	0.408040579	24.74983928	68.81444947	110.5953098	199.5410774	54800
’194126’	1.432726577	0.247610153	26.6814372	35.16546011	200.2533072	174.8904793	54800
’194268’	1.453079535	0.787367201	5.428755596	161.0896563	107.396251	50.6537459	54800
’196068’	2.117188183	0.664099337	59.40515062	33.67335375	251.1655594	29.5663475	54800
’199003’	0.959006581	0.152059963	21.61578726	288.0947674	297.5853194	239.8413611	54800
’199801’	1.684821679	0.569649652	2.28337519	245.8457318	86.51933458	33.88013061	54800
’1979XB’	2.351049444	0.726046355	25.13874324	85.49544485	75.7428619	2.936666778	54800
’1983LC’	2.613694398	0.708842552	1.519945709	159.5699088	184.836557	353.9391431	54800
’1988NE’	2.26871648	0.449864084	9.708569997	251.4703848	3.765629143	355.9835378	54800
’1989AZ’	1.647360204	0.468518899	11.78417015	295.6349798	111.8152084	170.5951022	54800
’1990SM’	2.101033091	0.766065191	11.57957264	136.8540175	107.0247606	9.579915699	54800
’1991FB’	2.370739218	0.566641274	9.037932418	18.06819952	219.9877987	287.587686	54800
’1991LH’	1.356770649	0.732577793	53.18574608	281.0876849	203.7472958	128.537589	54800
’1991TT’	1.19434631	0.161354453	14.80458918	192.4145699	218.0107167	22.52112329	54800
’1991XB’	2.940113852	0.589537108	16.30513453	250.3863387	172.3347861	129.4632735	54800
’1992BC’	1.422481343	0.352611682	14.36230282	123.4390308	77.09717786	296.4562433	54800
’1993BU3’	2.405610749	0.51441058	5.295749663	316.2591191	144.3178121	95.90536758	54800
’1993FA1’	1.426126024	0.288567104	20.45286483	187.3202529	343.5727844	82.54058062	54800
’1993RA’	1.918646327	0.416183834	5.5996968	171.9121477	264.9808472	215.0462297	54800
’1993UD’	1.319726346	0.194528077	22.79342512	25.08054354	254.8506876	79.45010641	54800
’1993VC’	2.775607624	0.532345641	3.206068164	242.4193446	177.1088741	85.79943313	54800
’1993VD’	0.8761973	0.551406373	2.062210141	2.720247353	253.6621141	234.1661206	54800
’1994EK’	2.158002585	0.639999967	6.049027623	333.2352081	99.05802105	259.0103535	54800
’1994GV’	2.009515607	0.519506473	0.457943032	20.1056358	154.0100241	56.87009092	54800
’1994UG’	1.238293494	0.29238149	5.20697655	13.75589969	225.4443341	217.0882987	54800

’1994US’	2.735411989	0.56826597	8.491427894	223.6616067	121.3912966	55.55009823	54800
’1994WR12’	0.756417685	0.398423118	6.871125181	62.85018111	205.8901481	235.2581666	54800
’1995CR’	0.906731886	0.868514097	4.037199387	342.7586676	322.4166786	293.0512482	54800
’1995EK1’	2.265143725	0.775752749	8.857172808	355.454349	296.817402	347.8011489	54800
’1995FF’	2.319499482	0.709407366	0.590570641	172.4707826	296.3640056	322.0561844	54800
’1995FG’	1.849264049	0.372840709	1.961807555	184.998189	36.75070841	142.8699337	54800
’1995SA4’	2.500845606	0.578734493	2.816679247	187.6644943	148.2776546	127.4990414	54800
’1995SB’	1.320207078	0.085431617	14.88881055	352.0248442	263.456165	341.4896121	54800
’1996HN’	2.203356303	0.411205201	8.605892734	203.0441918	25.13750265	301.9254521	54800
’1996TE9’	1.793128796	0.325911394	21.63630461	13.93824646	3.7813878	16.99465823	54800
’1996XW1’	1.725680152	0.454148444	30.59206717	247.9958073	264.519646	65.93491598	54800
’1996XX14’	2.549335465	0.649873055	10.55951874	195.5917571	185.0015455	351.2037157	54800
’1997CD17’	1.122613203	0.141564167	15.10417353	320.4837531	221.4337668	301.1266988	54800
’1997GL3’	2.281152788	0.782810282	6.688630358	196.2470794	260.5912382	153.3371724	54800
’1997MS’	1.936905147	0.727667016	54.96607793	86.33214307	65.83230523	112.2163103	54800
’1997QK1’	2.79967498	0.640195313	2.877547033	307.0777178	2.554461644	149.5519662	54800
’1997UA11’	2.363046438	0.62045902	3.300251178	212.5126053	138.4294636	26.17825764	54800
’1997UH9’	0.830094504	0.474683526	25.49129072	42.44034664	180.8619857	48.75397365	54800
’1997US2’	1.674009982	0.660636506	3.169584181	66.24658022	99.87673	357.3748758	54800
’1997US9’	1.052655845	0.281884202	20.016011	212.2612861	357.3262979	284.8632433	54800
’1997XR2’	1.076944541	0.201215856	7.172531662	250.8245182	84.58487206	13.66060782	54800
’1997XS2’	2.662154055	0.521022275	19.48431617	75.22249476	24.00076014	186.9348516	54800
’1998BT13’	2.457000162	0.596897592	1.415007722	123.3570673	353.3797913	293.4328724	54800
’1998BY7’	2.024221774	0.604498608	3.282234303	122.3655312	90.00692526	251.9315646	54800
’1998DK36’	0.69231541	0.415487023	2.02699123	151.125101	180.3693782	70.79126383	54800
’1998FN9’	1.396477869	0.235613273	14.6244169	183.9013271	329.2150123	192.0813957	54800
’1998GC1’	1.442352291	0.293128331	18.750722	19.12747339	117.0890437	85.62045628	54800
’1998HM3’	1.246510401	0.062186869	39.32826404	210.8468624	137.0067219	95.68617481	54800
’1998HN3’	3.11858015	0.618525946	9.217724306	49.17966399	250.727982	305.5343204	54800
’1998KH9’	2.203262383	0.458232285	17.66522372	82.73192373	184.9506904	67.73473891	54800
’1998KJ17’	1.986054397	0.482830067	9.148299684	76.07418649	163.9758447	273.8913009	54800
’1998KM3’	1.671466487	0.611295941	4.661821752	263.3984126	84.92864909	272.6349406	54800
’1998MW5’	1.022801173	0.362662274	6.287074577	80.47583114	26.65671591	172.2878231	54800
’1998QA1’	2.104380797	0.532169372	8.16501834	299.1460316	332.9167666	150.4237625	54800
’1998QA105’	2.700851892	0.533464999	8.327575247	337.0819727	39.83107683	100.4451557	54800
’1998QQ’	1.233177065	0.680136698	36.6954847	325.3822177	220.380832	246.967606	54800
’1998QQ63’	2.362497204	0.550481017	1.667655083	104.4642788	265.5239708	290.7411038	54800
’1998SE35’	3.018446371	0.589923896	14.72019832	334.1454422	43.78343684	340.4731834	54800
’1998SL36’	1.394277611	0.420040141	19.15816658	353.1694232	116.6301368	358.6094922	54800
’1998ST4’	2.814406307	0.599012284	9.300916153	239.3280533	207.0530196	26.06628985	54800
’1998SV4’	0.816494924	0.641998453	53.29453671	177.2616613	359.481833	99.19350085	54800
’1998UY24’	1.364007899	0.321219646	16.88437027	38.79876789	275.6140152	166.2518009	54800
’1998VD31’	2.651416697	0.803637427	10.23830094	47.70006344	113.3877918	102.1478779	54800
’1998WC2’	2.551903392	0.563218787	26.55146254	208.9225176	269.8490719	150.9343136	54800
’1998WP7’	1.212975809	0.427364054	21.62712516	230.914426	64.81608632	247.3687798	54800
’1998XX2’	0.741105994	0.367472879	6.969350195	74.55769826	152.833163	96.72474379	54800
’1998YF10’	1.490767779	0.247874059	15.61043376	85.98483133	296.4017263	203.2364475	54800
’1998YM4’	1.476211239	0.719694769	3.436554	341.8799877	344.3989273	230.1820209	54800
’1999AF4’	2.823465559	0.618914247	12.59241109	294.94861	154.7888469	31.4131741	54800
’1999AU23’	2.159898297	0.410376208	20.42509712	293.6980561	117.8809408	54.67090114	54800
’1999CW8’	2.237356026	0.597784558	33.65592361	317.1658652	262.0505015	313.3650043	54800
’1999ED5’	1.745519014	0.466373643	18.18065658	5.675118385	273.8844439	33.96109819	54800
’1999FP19’	1.942577471	0.51989595	15.0808404	347.0635868	273.9539411	180.5518567	54800

’1999GL4’	2.116341586	0.603389587	7.246971111	178.6730651	293.6740083	72.08451061	54800
’1999GR6’	1.348494559	0.763186695	29.31642065	180.9636408	146.469223	19.5244093	54800
’1999GY5’	1.146242751	0.614504957	24.44372701	203.4622906	232.1438501	358.8746838	54800
’1999JU6’	1.469095456	0.200660349	22.45740089	220.2216436	69.00416229	92.21480724	54800
’1999LD30’	2.852814514	0.622957913	8.420513665	86.17411686	205.5414073	333.93877	54800
’1999LP28’	1.219339414	0.090844754	16.31253316	88.00395273	306.9722615	236.5956421	54800
’1999LV7’	2.208461566	0.470663868	30.4802745	87.48366231	158.9259591	319.834044	54800
’1999RK33’	2.4864954	0.585922151	2.898494358	319.3882032	53.68080486	121.7744515	54800
’1999TA10’	1.505677161	0.241650289	20.84270601	214.7319767	84.68784438	35.57480722	54800
’1999TC5’	2.015031493	0.548180238	29.09760182	192.5272818	282.604299	37.89788551	54800
’1999TM12’	1.589171133	0.46434531	28.59641486	199.0500288	250.8500279	175.6130291	54800
’1999TN12’	1.886495229	0.391060478	37.25811658	212.3955957	150.2131756	205.2910339	54800
’1999TO13’	1.582267873	0.435729476	20.27521907	15.02177427	302.2695557	235.9462101	54800
’1999TT16’	2.164314679	0.665882512	2.014934144	328.2029962	148.6437252	279.8696772	54800
’1999TW16’	1.424845088	0.735595341	34.6294019	28.33824449	134.6623503	91.19539575	54800
’1999TX2’	1.28081798	0.463158289	61.38858117	179.9750715	53.56548743	185.7170889	54800
’1999VG22’	1.646608012	0.330069998	2.853425526	271.4645234	222.4794678	49.78363178	54800
’1999VM11’	1.594543985	0.279099333	17.05250545	218.090849	206.306954	164.9994467	54800
’1999VN6’	1.733287714	0.370498586	19.48471702	58.12750087	43.5607165	323.3276092	54800
’1999VR6’	2.211801718	0.761395808	8.567884843	213.0808032	294.0154261	254.4555579	54800
’1999VS6’	1.197569876	0.223080114	27.97163962	35.34448305	89.29411567	267.4662016	54800
’1999VV25’	2.230527196	0.555390193	7.431757438	231.8507388	204.6371794	253.4825399	54800
’1999VX15’	3.003014471	0.601954735	12.34010729	222.4049351	268.2908753	235.6433813	54800
’1999XM141’	1.238891608	0.370553874	21.68182803	73.11532536	105.5309005	132.763574	54800
’1999YF3’	1.48698521	0.143351242	26.76285041	298.9482593	147.07168	340.415903	54800
’1999YR14’	1.653651269	0.400692618	3.722193016	3.133896349	9.414387529	114.7340213	54800
’2000AA6’	1.289429601	0.520724878	2.046821823	280.2362895	287.2405274	345.6831727	54800
’2000AD6’	2.214009106	0.423661801	30.06113454	112.8698517	335.1563939	263.121297	54800
’2000AD205’	1.693236917	0.585854124	7.927731543	157.2792218	191.8936416	49.16875988	54800
’2000AE205’	1.164091641	0.137352457	4.459952011	271.7038089	150.3179304	65.76423875	54800
’2000AG6’	1.017661088	0.189842472	2.435327189	283.0958052	276.3479347	175.0141853	54800
’2000AG205’	2.291692135	0.521107063	18.41314678	269.9788861	249.00486	179.5095472	54800
’2000BK19’	2.413162296	0.576885432	14.78234434	310.6749897	200.7922512	124.0089011	54800
’2000BL19’	2.727676858	0.639655644	15.13060082	314.8235685	241.2049124	333.8171519	54800
’2000BO28’	1.698792435	0.599429597	6.338255315	320.0807351	303.0330986	299.9599469	54800
’2000CR101’	1.694740465	0.246209949	0.599457594	333.7572045	152.9220412	8.698136827	54800
’2000DN1’	2.883839566	0.669219115	7.772648953	42.3468173	146.1906671	278.0911933	54800
’2000DV110’	2.091269766	0.387190882	4.395924775	11.67358759	220.2593281	280.3801818	54800
’2000EA14’	1.116850414	0.202516088	3.554314075	203.9815847	206.0596511	236.6728054	54800
’2000EB14’	0.895576549	0.495441881	11.56195226	162.8952775	139.5771723	26.14335771	54800
’2000EM26’	0.816086953	0.469756898	3.873103112	345.2547298	23.96570784	66.77439243	54800
’2000EU70’	2.226629623	0.76599184	13.07704528	166.2399094	253.1626193	249.6290114	54800
’2000EV106’	1.648623884	0.348672951	33.47449983	170.9230192	255.2998169	107.5776926	54800
’2000FP10’	1.440085934	0.237435915	23.57414594	5.13749573	165.2960095	14.39266447	54800
’2000GC147’	2.771938652	0.615179958	2.327840633	319.0680723	126.8503	319.6391936	54800
’2000GF2’	1.340968048	0.377572644	9.627538303	176.1226664	107.9897087	161.879806	54800
’2000GV147’	1.746532508	0.456277961	10.57183715	68.76523129	215.9375377	237.4029065	54800
’2000JB6’	1.786449986	0.345339087	10.63996081	110.7083993	175.4648853	178.8016813	54800
’2000LF3’	2.579260041	0.660010689	14.94730921	83.19810531	222.6302202	4.160561969	54800
’2000LG6’	0.917369681	0.111128708	2.830919257	72.57085867	8.089670403	54.99053768	54800
’2000OG8’	2.667743712	0.542252985	5.286942821	295.9247209	70.83784438	314.0732954	54800
’2000OH’	2.423161545	0.590323347	18.56138531	284.0805373	354.5093876	83.01985822	54800
’2000OK8’	0.984679929	0.221126738	9.985597025	304.6319964	166.1394703	38.91888985	54800

’2000PD3’	1.998675458	0.592745904	7.689197348	299.0111792	109.7771231	307.731591	54800
’2000PG3’	2.824759665	0.859259445	21.64787227	324.1859574	140.8075134	241.2147106	54800
’2000PN’	1.01964328	0.761326324	22.77552587	128.7671924	37.7417606	85.06642193	54800
’2000PO30’	1.835803753	0.399719099	3.578645943	201.1349943	217.8569126	110.9528696	54800
’2000PP9’	2.327511183	0.553111991	5.59133109	171.7746159	159.4724908	119.9894871	54800
’2000QO130’	2.249609131	0.455610656	5.900361024	343.2656392	339.4863103	166.0749258	54800
’2000QU7’	2.279165965	0.648935438	22.32782591	339.6029497	87.34829753	123.9625989	54800
’2000QV7’	1.408677651	0.522910306	9.125016136	154.9007741	79.36673502	18.99498076	54800
’2000QW7’	1.946293114	0.467928316	4.162681062	158.7301825	190.5935352	10.08133732	54800
’2000QY69’	1.426536617	0.163689299	25.20357364	342.0267051	64.96141553	251.9047387	54800
’2000RD52’	2.203947646	0.443865658	4.945601573	309.1477827	24.36908001	188.0236242	54800
’2000RJ12’	2.152288024	0.482218527	7.150137335	318.0970041	25.67905939	218.1626992	54800
’2000RK12’	2.519214879	0.68706092	31.82270189	161.6258689	251.4508465	9.504514264	54800
’2000RK60’	2.178196414	0.488089703	6.585929981	197.2056648	133.4113652	206.5848575	54800
’2000RN77’	0.951002945	0.318273452	16.0958665	312.8411469	211.7109183	142.8437694	54800
’2000SB8’	2.277946518	0.480209146	8.777081639	43.01783824	311.9517694	140.4065568	54800
’2000SN10’	2.494700605	0.549599221	11.7603497	353.4658308	353.2208681	31.6057149	54800
’2000SS43’	1.731818313	0.409779332	4.093698838	6.896856145	353.1915483	212.9904319	54800
’2000SZ162’	0.930171312	0.167694558	0.893353596	14.82238438	131.384934	275.7039381	54800
’2000TE2’	1.3204527	0.213922313	6.220658041	9.158973883	9.944158731	131.3141143	54800
’2000TG2’	1.521606455	0.245328089	11.99889796	206.8983252	200.3110272	98.2565091	54800
’2000TH1’	2.307924346	0.541990586	12.13108439	12.79618677	335.2891472	124.4531144	54800
’2000UG11’	1.928400337	0.57294023	8.924763172	224.2714489	240.5319015	348.0963372	54800
’2000WH10’	2.52613563	0.663057035	13.26211028	50.22042447	63.36425855	349.3101187	54800
’2000WM107’	2.541885342	0.60725575	19.35172519	71.65440284	280.9760329	12.03563637	54800
’2000WP19’	0.854472024	0.288621909	7.681142545	55.83243571	222.0306299	170.4192779	54800
’2000WP148’	1.318294513	0.264170303	21.28130817	252.0690351	237.8394727	58.19201345	54800
’2000WT28’	2.551925921	0.610373815	5.654897774	47.10176858	19.79509457	343.2507671	54800
’2000WY28’	1.63861427	0.289485311	19.44771735	63.36201341	357.2867791	292.8550262	54800
’2000YG29’	3.174964828	0.692976349	18.88808716	92.44044174	358.491086	147.7902758	54800
’2000YJ29’	1.961315005	0.833369084	43.68747752	266.2201375	327.7354496	294.1130798	54800
’2001BA16’	0.940314425	0.137423212	5.76889984	115.6154682	242.8093806	330.7986698	54800
’2001BE16’	1.255021329	0.40399654	39.38282642	121.9705856	106.147857	148.2137999	54800
’2001BN61’	1.82803812	0.464215152	9.731194897	118.8962583	334.1645257	73.00729238	54800
’2001BZ39’	1.988437523	0.421272478	8.834776278	225.1741304	197.0207766	318.1460054	54800
’2001DB3’	2.685154738	0.558752686	24.5629739	341.4982933	263.520601	243.6793877	54800
’2001DC77’	2.543401331	0.500017004	9.404034044	356.321546	195.1042739	318.8552096	54800
’2001DQ8’	1.841905584	0.901427445	12.89105188	342.8591765	14.59819642	68.93618794	54800
’2001DZ76’	2.356516978	0.608636603	5.750598301	152.1215452	38.11110915	45.99200895	54800
’2001EC16’	1.345509649	0.363932403	4.711455201	175.626489	70.39778891	303.413813	54800
’2001FA7’	2.007041543	0.535896744	22.85874486	352.5952239	62.39915481	316.3529175	54800
’2001FP32’	1.359490362	0.333559785	29.29196932	182.5790748	64.70258151	272.7271582	54800
’2001FX9’	1.932329589	0.33142956	3.494136644	184.5507318	291.9119833	344.3139501	54800
’2001GO2’	1.006559674	0.167995166	4.61482654	193.5925501	265.3671952	288.7554105	54800
’2001GQ2’	1.213857966	0.503066548	21.82149385	37.21415429	280.2228046	201.2996833	54800
’2001HA4’	2.684549923	0.795805012	17.16345089	355.0036277	94.70615782	275.0697216	54800
’2001HA8’	2.383634668	0.530160564	11.54309261	96.03406244	201.8300045	351.2434748	54800
’2001HC’	0.874652993	0.499269928	23.74189495	32.64190441	28.16788172	221.8875507	54800
’2001HX7’	2.261303168	0.507679101	56.75771004	205.4758634	40.67101163	74.37508159	54800
’2001HY7’	0.913901433	0.411990202	5.209114436	205.369114	211.0087779	26.12750726	54800
’2001HZ7’	1.468368332	0.498244867	5.41699694	156.4726297	312.6188791	142.3376312	54800
’2001JW1’	1.178348932	0.068750718	35.39315184	62.06815275	97.41877021	31.74122852	54800
’2001KD68’	2.381339193	0.505520136	2.043518083	249.8508171	350.5200046	16.92368435	54800

’2001KW18’	1.241935786	0.157212084	7.188106329	61.71019444	181.8309128	155.3518391	54800
’2001LC’	1.054476756	0.677552707	16.97028858	112.4273797	2.42084235	30.63061417	54800
’2001LM5’	1.231557369	0.034624715	12.54781974	270.2430162	255.7891812	263.58421	54800
’2001MR3’	2.364408249	0.454573167	4.444779909	221.0541799	58.43056739	14.45123263	54800
’2001OA14’	1.089129792	0.421622836	29.23969107	309.5655804	144.3381902	65.89708551	54800
’2001OD3’	2.61071514	0.520849279	14.94314311	122.5177001	209.9939666	259.4320714	54800
’2001OX13’	2.382698	0.462469076	4.1833437	54.52005475	277.4226993	349.2896303	54800
’2001PD1’	2.235250658	0.456966374	5.961092811	282.5324574	94.49719805	51.94990351	54800
’2001QB34’	2.20619469	0.417451269	5.736760866	267.1086916	86.2148165	76.18000572	54800
’2001QD96’	1.274386799	0.496610763	17.95303947	330.367458	145.7208677	260.8496757	54800
’2001QH142’	1.527460085	0.221646392	30.60080644	318.3848612	253.4377061	64.37117675	54800
’2001QM163’	2.320364548	0.73450184	9.227150037	88.02739236	165.8495727	34.82324868	54800
’2001QN142’	3.089031686	0.686167982	10.23400212	164.3801305	110.1665775	131.4211186	54800
’2001RA18’	2.6027851	0.591959851	10.98173862	179.7559227	205.6723794	247.6419571	54800
’2001RP3’	2.345062855	0.558817351	9.269959219	170.6984893	158.622759	6.761725291	54800
’2001RQ17’	2.003259322	0.492798503	1.33007288	30.85767009	284.3442889	206.4018735	54800
’2001RX11’	2.771017168	0.544080298	13.04801686	344.0105391	307.6154175	216.9851986	54800
’2001RX47’	2.021439103	0.422953095	10.745862	277.0742631	23.69117354	201.1816247	54800
’2001SA270’	1.302336577	0.735284528	38.53904347	210.056282	15.56137746	331.7898017	54800
’2001SD348’	1.884914509	0.328349226	14.38562097	202.9330964	89.36974513	318.3580949	54800
’2001SE270’	1.215050219	0.481848872	5.369229404	357.0874808	107.5412393	83.12727017	54800
’2001SE286’	2.035426097	0.456871455	26.85193065	268.6461204	199.033326	135.7679739	54800
’2001SG262’	1.963510461	0.582482272	4.810999194	359.6434776	99.48799462	196.6339845	54800
’2001SQ3’	1.110197381	0.254518561	23.89749164	356.2138636	268.1059758	121.1600771	54800
’2001SS287’	3.247082196	0.674199525	18.41289784	230.8608013	173.8492978	76.69844682	54800
’2001SY169’	1.227015052	0.408218196	5.098072644	171.6503399	82.01965709	163.5684074	54800
’2001TB’	1.716852637	0.525264501	3.96803148	192.2697714	245.0530232	44.12379043	54800
’2001TD’	0.954140682	0.166064739	9.012583297	13.2106394	241.3570728	343.2961109	54800
’2001TE45’	1.805816098	0.462799927	14.60699116	208.7424417	120.3518838	355.6720236	54800
’2001TY1’	2.407932814	0.593216507	5.849968171	9.695905912	342.8256886	332.4775229	54800
’2001TY44’	2.361456421	0.51875975	2.535546875	357.1802301	72.87966257	333.0755081	54800
’2001UC5’	2.731184153	0.625179892	30.39936453	28.36207463	23.90710034	202.8922718	54800
’2001UE18’	1.361370965	0.181932518	15.44863116	33.23576633	60.63840275	125.4059822	54800
’2001UF18’	1.141145475	0.609493396	31.27882849	46.02419845	202.5310502	7.456045204	54800
’2001UO’	2.549019692	0.66901848	10.27737818	21.76582119	303.2269582	278.4235918	54800
’2001UO27’	2.631718779	0.520976543	40.19693732	217.1342885	153.0222066	242.3279709	54800
’2001UQ163’	2.17154795	0.492902769	6.757442667	56.08827079	262.3254046	108.936898	54800
’2001UW16’	1.36409284	0.178314209	37.5980402	37.07807922	106.6571113	72.34735334	54800
’2001UW17’	1.557912892	0.232207493	12.90404025	223.2987375	217.7671276	200.9569742	54800
’2001VB2’	1.718264156	0.395329397	7.93570965	50.1396775	319.5865391	57.52201342	54800
’2001VB76’	1.45840335	0.348422792	4.238976233	259.5823245	248.2432802	302.797615	54800
’2001VC76’	1.754545551	0.442167828	16.84997651	52.34658652	258.6994651	60.6864087	54800
’2001VE76’	1.741008131	0.514638833	4.155557011	217.3560133	262.461042	1.612754126	54800
’2001VF2’	1.818041065	0.384620898	8.910993997	58.89839992	11.66779833	308.8394828	54800
’2001VH5’	1.273963264	0.186643694	26.51610168	230.4481215	119.3220549	6.189209693	54800
’2001WH1’	2.466794363	0.800362751	15.57988302	68.33207057	107.2929653	276.1030339	54800
’2001WJ4’	1.255446138	0.21637657	7.908879629	57.22819474	18.6522551	347.4227276	54800
’2001WK15’	1.140978336	0.136962138	24.50933226	66.3961333	99.89068343	189.5983638	54800
’2001XE1’	1.603518778	0.210851683	20.95146854	231.1865884	272.9687611	107.2855405	54800
’2001XF1’	1.478885427	0.463849001	22.0408131	87.87359182	231.480811	14.99911158	54800
’2001XG1’	2.006133215	0.598132588	3.017506561	56.5245504	81.36252579	149.8963753	54800
’2001XP88’	1.346800191	0.194436491	6.74825863	97.90598791	261.1695601	229.2930137	54800
’2001XQ31’	2.850458554	0.578046574	19.51189364	85.70161983	333.5346284	168.9053558	54800

'2001XV10'	2.206723375	0.583640463	22.27384465	31.54259549	341.7853554	82.10803375	54800
'2001XW10'	2.103840397	0.767626065	4.65904182	340.4833282	353.5362551	121.7546738	54800
'2001XW266'	2.286721743	0.449217817	4.617255121	206.1975157	89.40220896	36.71276531	54800
'2001XX103'	2.051599716	0.665344781	6.171876559	79.8066488	84.81120073	115.9114909	54800
'2001XY10'	0.871777972	0.387232628	30.99519754	92.9782006	219.6724533	308.0723486	54800
'2001YA1'	2.161296127	0.472766122	27.97899194	255.3269406	167.91376	71.28357537	54800
'2001YE1'	1.912031704	0.502088267	4.457827872	65.92907998	97.54430972	200.4746119	54800
'2002AA'	1.147936608	0.302333965	11.27285637	302.2687327	64.5563748	280.8413397	54800
'2002AC29'	1.641601873	0.503288006	26.58007334	85.21069795	129.5424309	22.84367944	54800
'2002AF29'	3.246806432	0.614649403	8.001787483	217.2108243	282.5070126	61.03104462	54800
'2002AU4'	0.855563804	0.373648298	17.18086593	99.50935788	205.1518748	70.17955517	54800
'2002AV'	2.465190371	0.660778977	2.841110632	124.0794251	285.5176274	292.7899006	54800
'2002AW11'	1.44696024	0.330393024	18.33071146	95.63424888	91.03062825	316.2769354	54800
'2002AY1'	0.778708828	0.437707559	29.88734905	287.8976428	323.8569822	260.0228131	54800
'2002BJ2'	2.048472888	0.653679482	26.13153875	16.65232926	60.852933	154.9107556	54800
'2002BK25'	2.296507125	0.749096098	11.92852279	156.4002852	103.6926275	321.819486	54800
'2002BM26'	1.832228657	0.445039301	16.2272804	319.6957722	180.4012074	265.996196	54800
'2002CS11'	2.020229504	0.4032339	9.803541658	346.3483434	160.1667661	132.3610208	54800
'2002CT46'	2.361227912	0.531198086	15.73443638	157.4711245	356.2052351	312.3663391	54800
'2002CT118'	1.279336658	0.350852016	10.37527921	321.9528585	266.0906642	205.0504321	54800
'2002CU46'	1.737600391	0.565282708	32.18865823	145.1966894	92.75510267	318.0776315	54800
'2002CW11'	0.865480941	0.225580829	3.133939157	137.6247826	210.350877	301.4838274	54800
'2002CX58'	2.797331811	0.659214558	2.534185251	110.1827277	75.74394095	154.4807085	54800
'2002CY9'	1.648573998	0.508439149	41.9709023	305.3823426	117.3725943	103.4987906	54800
'2002CY58'	1.365977733	0.384396922	8.283664232	341.6895742	39.94642982	168.9988095	54800
'2002CZ58'	2.187921554	0.465408233	18.82845986	339.5615337	87.9280756	62.27078937	54800
'2002DJ5'	1.400369221	0.567800104	6.439700302	348.099162	295.9973965	311.8300734	54800
'2002DQ3'	1.387188679	0.254895052	5.051837003	340.6195642	160.1812448	58.63184399	54800
'2002EB3'	1.758104388	0.684453917	9.912684731	1.791811501	300.0324237	283.5292734	54800
'2002EM7'	0.92123263	0.36297078	1.547432464	347.2071486	57.68837704	302.9851282	54800
'2002EV'	2.45601064	0.581939385	10.41758245	156.310531	353.4805176	273.3206157	54800
'2002EY'	2.136692846	0.630390143	20.23174164	170.8846787	264.8570546	77.97823316	54800
'2002FB6'	1.796476087	0.54493341	33.70223562	182.8208581	101.775922	254.485454	54800
'2002FW1'	0.823350353	0.342015469	6.592641824	164.1464448	223.1410409	109.6715861	54800
'2002FW5'	1.314970529	0.217718142	46.44431137	21.77561836	85.42704348	221.637853	54800
'2002GF8'	2.283567733	0.453706458	4.886860254	20.37466094	237.1577162	313.2294503	54800
'2002GK8'	1.841243135	0.401400292	38.99834189	47.89325252	266.7539684	182.0323429	54800
'2002GM2'	2.198649831	0.807975231	3.356043697	339.8801188	83.82749971	42.83499017	54800
'2002GR'	1.201532225	0.20760734	7.323664581	183.8431858	313.6817245	56.86827116	54800
'2002JA9'	1.985011041	0.483084906	10.51024232	93.72876597	230.2780689	79.45580851	54800
'2002JB9'	2.71823306	0.784872235	46.73548604	70.42208228	277.9028065	145.1819748	54800
'2002JE9'	1.067779968	0.416720917	8.827300453	200.1437533	255.3844115	58.91935416	54800
'2002JQ100'	1.180542737	0.518847295	29.6089218	47.37797396	40.06189048	124.9106473	54800
'2002JR9'	2.38608516	0.636909651	9.907490845	122.9549295	203.4021088	257.9600591	54800
'2002JR100'	0.924673959	0.297798077	3.763163526	203.5554627	253.4473863	238.742324	54800
'2002KJ3'	2.268790014	0.488790857	6.425884385	48.12577792	252.0640164	302.9323481	54800
'2002KL3'	1.951566445	0.74907457	20.72596584	73.50185983	293.0282404	106.7479413	54800
'2002LZ45'	2.310275074	0.634019152	6.186032263	91.34935757	239.7583351	287.8762509	54800
'2002MT1'	2.37469308	0.497596887	4.730939436	205.6823638	82.46721368	268.4317805	54800
'2002NA31'	1.665988794	0.288598273	19.97699187	125.5706689	219.7839621	321.0840709	54800
'2002NW'	1.608708734	0.667970403	6.027175137	102.3808595	287.9878283	22.19775304	54800
'2002PE130'	2.558271889	0.619640438	15.62659015	357.5407176	33.46477303	180.9463446	54800
'2002PH80'	2.169831897	0.458211314	6.422528235	351.3040292	331.1878721	351.6834083	54800

’2002PO6’	2.235006677	0.5190652	20.59061367	304.3291629	301.1681461	344.0680949	54800
’2002QE7’	1.469822183	0.18118064	12.11033232	244.5516752	88.33742076	180.692713	54800
’2002QH10’	2.360933278	0.56042801	4.793405489	0.483469669	23.71232485	248.4493451	54800
’2002QQ40’	1.215227133	0.564610745	1.723280561	104.1637021	356.4549576	183.4716136	54800
’2002RC118’	2.951243209	0.566205912	28.03480978	208.9828054	222.2018622	68.94707347	54800
’2002RH52’	1.978928668	0.492282161	16.18692263	2.371189429	96.61246354	39.71939245	54800
’2002RP28’	1.66196005	0.390539866	7.999730006	163.5247819	250.4604408	295.2897787	54800
’2002RT129’	1.832369738	0.75378	19.63694368	178.725369	55.64949635	207.5664476	54800
’2002RV112’	2.220682998	0.489130312	16.50556811	196.1230305	199.3967504	313.4148508	54800
’2002SL’	2.201788042	0.499294087	6.505614043	139.2924772	151.955489	348.6953554	54800
’2002SN’	1.949039648	0.377894729	6.42373428	36.60968638	337.2152128	93.54011043	54800
’2002SQ41’	2.6047404	0.801892554	25.06966704	22.71579201	95.12030107	153.3768165	54800
’2002SR’	1.179824011	0.196038964	6.688557057	160.9435121	285.0759198	237.3294474	54800
’2002SV’	1.403351061	0.236890694	16.7732143	352.7117301	326.8581254	280.5438713	54800
’2002TS67’	2.345431522	0.491380059	9.697026954	79.52333193	322.8100884	248.8607967	54800
’2002TX55’	2.228310482	0.570864906	4.379631401	190.2452352	148.8380891	313.0404183	54800
’2002TX59’	1.223476724	0.145773336	12.97641826	188.5875741	132.7027374	234.4470395	54800
’2002TY68’	2.218097532	0.514100119	20.80169465	19.91474498	57.98729072	282.906895	54800
’2002UL11’	2.811703709	0.558524122	9.743318037	52.39453254	331.9474222	104.3657037	54800
’2002UN3’	1.744143314	0.257344179	8.695680562	28.08034346	112.5961626	155.7827621	54800
’2002UX’	1.473432334	0.163396909	20.20657942	263.9213204	84.27950545	179.534241	54800
’2002VD118’	1.427632787	0.143647327	14.25044998	35.26763358	66.27651216	155.8687947	54800
’2002VE68’	0.723589662	0.410515769	8.980538881	231.6524629	355.5027705	121.1010776	54800
’2002VO69’	1.44044533	0.349151656	20.94255275	47.35438863	55.67914894	152.4600747	54800
’2002VQ14’	2.584349693	0.510503036	7.228831592	236.4274501	159.056878	167.5041802	54800
’2002VR14’	1.625655533	0.498835067	5.532247969	76.32103523	44.11651262	304.74967	54800
’2002VV17’	0.83743279	0.43653465	9.698213272	222.2942569	348.7650752	196.7992274	54800
’2002VX91’	0.985069444	0.201283632	2.335509572	216.7237513	78.10746459	160.1639145	54800
’2002VY94’	3.241663611	0.658221007	9.152408967	280.9452523	233.1949082	339.605983	54800
’2002WP’	1.449839876	0.215919394	19.15205651	76.37420823	1.020911412	145.1069174	54800
’2002WX12’	1.750391641	0.659555541	8.634530064	211.7431658	328.2909805	183.0926165	54800
’2002XA’	2.827125889	0.626061226	3.319145604	96.13151487	34.80338322	80.71581713	54800
’2002XA40’	2.263091915	0.481689719	4.455688695	300.9527034	66.45249608	311.3577047	54800
’2002XB40’	1.854174333	0.553654132	6.800386455	255.6920319	118.1569045	149.3992435	54800
’2002XM90’	1.79108251	0.377919314	20.18254382	81.70633578	80.68027989	140.2030278	54800
’2002XN14’	1.766465822	0.440553705	11.78561414	86.27957247	310.7453362	216.3242928	54800
’2002XT4’	1.609844263	0.422539507	7.508260624	67.35116035	313.1270337	355.8648577	54800
’2002XT90’	1.02977307	0.223789924	43.38818726	287.699753	43.05087977	354.0745161	54800
’2002XV90’	1.578813707	0.376262798	9.994957521	79.03541887	356.2742127	3.416435591	54800
’2002XX4’	1.789138759	0.283945668	25.60576878	74.15572688	286.9544726	224.132252	54800
’2002XY39’	1.447933301	0.167117265	21.44647552	252.3195567	213.7571547	134.7586056	54800
’2002YD12’	2.21957798	0.49951012	14.72561509	276.0331896	193.0803267	279.1830074	54800
’2002YO2’	1.500730676	0.295114293	6.438108116	278.7805694	187.5991197	73.05064292	54800
’2002YQ5’	1.286947194	0.123548131	15.54667559	279.9840715	258.5080245	314.012964	54800
’2003AA3’	1.421295162	0.289405602	13.78222211	106.4551926	346.9199795	177.6803785	54800
’2003AA83’	2.493155935	0.783405003	6.84508659	88.58985213	126.4449714	163.2835599	54800
’2003AF23’	0.874850924	0.426196628	23.23720276	286.8241785	43.94682552	166.921073	54800
’2003AO4’	2.165591071	0.451913099	21.08122053	306.5358488	112.0814115	326.7910193	54800
’2003AS42’	1.501244604	0.341262152	9.982372616	107.4223301	24.0669204	62.16366416	54800
’2003AY2’	1.821649115	0.564803013	10.30753671	275.3312121	285.5597592	116.1774032	54800
’2003BA21’	1.100298773	0.83313449	23.73627108	308.9249898	18.07928025	105.372189	54800
’2003BB21’	2.228502361	0.556726555	4.845458564	344.8844864	211.6669374	254.9302624	54800
’2003BK47’	2.743225529	0.707944605	20.75098226	139.1570897	234.9525359	141.3849277	54800

'2003BN4'	1.269265789	0.170826571	5.600352111	307.7243202	192.8256574	22.27573824	54800
'2003BO1'	1.328496507	0.080582501	13.69613208	140.8526496	2.825319947	281.0238373	54800
'2003BS47'	2.099853678	0.521295083	4.886017965	131.6059637	358.2265642	330.2281192	54800
'2003BX33'	1.181968279	0.422627305	7.920504954	143.3415502	221.1164672	274.4757424	54800
'2003CA'	1.379095858	0.719532863	21.17866341	126.3355496	234.6328101	256.6184807	54800
'2003CR1'	1.453457248	0.463150126	12.71439116	311.1013129	101.8465117	156.6969553	54800
'2003DE6'	1.699445234	0.32858403	23.199593	160.1943215	68.65993565	172.594744	54800
'2003DF6'	1.220693674	0.17265946	25.784656	157.2538195	76.35971113	39.06760132	54800
'2003DN4'	1.145292852	0.477299662	36.30657697	157.8323437	141.5742695	157.6818026	54800
'2003DW10'	1.446357342	0.360772208	2.196658067	342.2652455	220.9388357	90.32828143	54800
'2003DX10'	1.375523716	0.410696309	3.147596208	61.97686519	193.7902099	152.2065781	54800
'2003DZ15'	1.220515647	0.486834037	3.651474461	142.013642	263.5714838	151.8710708	54800
'2003ED50'	1.41601913	0.546803534	33.72998511	173.611099	93.92992052	106.0362132	54800
'2003EG16'	2.399907371	0.686153443	20.27027055	152.1243124	109.8593685	177.8595411	54800
'2003EJ59'	3.21017086	0.620523116	13.41999895	0.386275049	167.1958009	357.7377765	54800
'2003EZ16'	1.175872726	0.139631519	5.804382236	341.9801435	71.45515575	279.3582656	54800
'2003FB5'	2.51152223	0.788524134	5.334906845	358.2716111	288.521537	141.0530596	54800
'2003FF5'	1.368596528	0.30343907	6.356351112	192.9916283	54.08792559	162.3962494	54800
'2003FJ1'	2.173591178	0.815481914	20.87798202	128.724437	181.7725616	256.9916558	54800
'2003FQ6'	1.371112171	0.131556728	3.620519905	86.6602047	233.3438258	72.30766502	54800
'2003FT3'	2.670619283	0.572445892	4.323888445	182.136911	84.10619685	87.76455935	54800
'2003FU3'	0.858495795	0.394042873	13.04100014	21.64755958	339.2687556	253.3751676	54800
'2003GD'	1.604472515	0.367321251	31.3542356	13.49030204	160.4239844	289.6883081	54800
'2003GD42'	1.294936307	0.228981323	12.24783454	18.17147062	156.1539974	313.7225937	54800
'2003GP51'	2.153051942	0.602977122	2.859325752	141.8415053	357.5426107	296.5506813	54800
'2003GS22'	2.739788158	0.577487103	2.420376157	148.0676231	4.365229386	100.0016448	54800
'2003GX'	1.329891379	0.207156376	10.85048911	19.63279552	167.9960373	251.5264757	54800
'2003HB6'	2.700813602	0.575316425	6.312596017	164.1740444	142.1579383	61.59932068	54800
'2003HF2'	1.113529941	0.675384261	3.056417463	190.0495824	230.851665	11.29414432	54800
'2003HP32'	2.697251963	0.777914277	3.432506839	188.0527314	155.4273681	67.30914104	54800
'2003HR32'	1.748354622	0.687468596	8.290173492	342.0380728	352.5540903	111.4826475	54800
'2003HW10'	1.797314642	0.532584664	3.916613843	37.88804984	239.2318036	97.99663384	54800
'2003JD13'	1.681270631	0.2717239	40.82595837	226.1638336	351.5989101	204.7027458	54800
'2003JD17'	2.457574645	0.552025747	21.09052067	219.5312081	34.16705088	154.5606448	54800
'2003JO14'	1.22491423	0.340638532	7.101208471	50.02999557	95.7008956	82.56673008	54800
'2003JV14'	1.633656972	0.579343004	5.463047159	99.1089802	33.18240437	270.9574105	54800
'2003KU2'	2.695895001	0.675282593	5.403717919	105.4569517	246.5370187	56.91304782	54800
'2003KX16'	1.334681666	0.579072431	23.63907989	94.3570823	28.23125643	262.1410348	54800
'2003LH'	0.960513471	0.149746576	10.79561453	247.325235	238.138055	51.78485912	54800
'2003LN6'	0.856893998	0.210516927	0.632617618	215.7369228	210.512885	162.6117511	54800
'2003LS3'	2.65432185	0.524063426	9.525623521	158.1215506	175.1028169	74.90307061	54800
'2003LW2'	1.877480775	0.479196794	2.294348229	247.1935258	335.2993201	58.3895847	54800
'2003MD7'	1.4670422	0.594344514	5.787454213	200.5865827	185.1602428	335.4913655	54800
'2003ME1'	1.040478329	0.302863161	21.05562691	268.6699854	255.5712851	116.2102183	54800
'2003MH4'	1.963015732	0.51471915	3.874243081	260.037953	322.8832206	10.32951594	54800
'2003MT2'	2.68227474	0.535798568	27.92010101	305.1857409	304.6266678	100.8904555	54800
'2003MT9'	2.536520826	0.92099956	6.824907875	233.3562645	200.6439985	110.8044846	54800
'2003ND'	2.220531876	0.450615391	5.115737627	297.2536553	23.15665472	221.2305609	54800
'2003NO4'	1.715056681	0.312860956	22.6977069	135.4315521	171.0627353	140.9892125	54800
'2003OQ13'	1.302575556	0.160784904	16.15040938	123.7933279	152.8077713	238.557141	54800
'2003OT13'	1.173573726	0.171778338	13.11783964	135.6191986	242.4436701	18.09792844	54800
'2003QC'	2.572787333	0.531249993	7.857623759	321.744673	37.39943961	92.59231544	54800
'2003QW30'	1.053445007	0.331782863	11.2766464	164.6189797	303.0843903	223.8107399	54800

’2003QY29’	2.634299311	0.59047789	16.4236187	326.9175639	348.6195856	86.76887943	54800
’2003QZ29’	2.479349449	0.602249998	8.717834533	162.8563058	251.0982954	104.3285889	54800
’2003RD5’	1.398364564	0.246880567	8.720066684	155.8629271	170.1262767	70.1376513	54800
’2003SF’	2.165272944	0.777655377	5.64519925	77.10649159	32.51672406	209.9072596	54800
’2003SG170’	1.858878016	0.604426976	36.94152449	199.4110602	309.1481527	318.4768173	54800
’2003SH84’	1.700157818	0.315398849	2.322690079	268.7840383	134.5622929	100.2144562	54800
’2003SJ84’	1.737789472	0.284245824	36.64687954	183.1274065	307.0079903	351.1339738	54800
’2003SK215’	1.851551538	0.446530556	36.01078545	2.854170923	288.9337634	46.91733934	54800
’2003SM215’	2.102508828	0.561698605	4.987840618	5.898342873	40.94228936	240.4432576	54800
’2003SN214’	2.036141752	0.549064381	8.332546098	353.5306498	284.5376943	306.5622333	54800
’2003SQ15’	1.665676918	0.458532442	45.3667958	333.992699	327.9086264	169.4036588	54800
’2003SR84’	1.706545962	0.476410651	7.53073234	183.424733	229.6962203	99.0867368	54800
’2003SU84’	1.498928013	0.385506985	16.46039091	181.9111559	95.48096544	338.4802798	54800
’2003SV159’	2.162558492	0.519372553	5.015651798	200.1416496	135.6219972	233.5697105	54800
’2003SW130’	0.883792255	0.30402724	3.667674513	176.5263837	47.73710981	188.9223263	54800
’2003SY4’	2.477617914	0.611854235	3.922136519	353.8143278	327.6477234	127.1154221	54800
’2003TH2’	2.451450815	0.670535335	1.3923293	50.30395942	44.31845689	107.6796543	54800
’2003TK2’	2.343077664	0.650337548	4.293055372	1.047113308	320.8781799	168.5936112	54800
’2003TM1’	1.361978655	0.562834868	1.704450449	271.5369466	354.8571658	128.5263746	54800
’2003TN1’	1.432811179	0.135802506	19.23228742	13.72438283	54.55246404	326.9586343	54800
’2003UB22’	1.219733034	0.225330699	15.8595352	212.4611246	116.1334468	323.4565251	54800
’2003UC5’	1.185347249	0.818281907	36.8429535	31.01446611	210.5903016	26.30507635	54800
’2003UC20’	0.781286551	0.336784955	3.794729949	188.8641335	59.28881366	267.8887041	54800
’2003UE22’	2.338576017	0.547148468	8.120486812	29.43924618	356.2068404	153.3046366	54800
’2003UO25’	1.016432467	0.228765904	15.4736231	211.9060116	28.51887453	127.5840364	54800
’2003UP12’	1.785254762	0.49074204	13.89929144	208.3507823	231.6676757	33.81441859	54800
’2003UP24’	2.236549815	0.49675152	21.68460879	213.2579784	237.6617391	169.1936671	54800
’2003UQ25’	2.534975202	0.681013728	2.126962806	187.4386431	276.645317	82.77590648	54800
’2003UR12’	2.431334654	0.568167165	60.46870576	194.0332197	122.3574366	137.3448414	54800
’2003UX26’	1.154745649	0.365308477	4.544726134	35.36829546	263.9817911	91.27795587	54800
’2003UX34’	1.095218711	0.615697927	2.566365109	4.676058173	218.1698829	293.3227326	54800
’2003VE1’	1.94275275	0.4950409	16.31114522	29.67864593	323.0498547	333.9893126	54800
’2003VG1’	2.699022028	0.575480259	8.807152789	331.8807372	135.2778389	39.00797578	54800
’2003WB25’	1.785050027	0.288630211	29.7974039	260.0401363	210.8776708	15.64276784	54800
’2003WE157’	2.249908793	0.565562448	9.621645424	227.3240408	263.6257589	155.0237187	54800
’2003WL25’	2.398467641	0.740800611	23.76678955	267.1780991	24.97948578	167.8642302	54800
’2003WO151’	1.545012746	0.663186846	19.79241692	228.1561669	330.7736394	158.3548382	54800
’2003WP7’	2.293812794	0.642487588	1.012921537	252.4797937	236.6534966	147.1602085	54800
’2003WP25’	0.991222895	0.121059601	2.528085679	41.85153566	224.6416742	177.6428317	54800
’2003WR21’	1.119082533	0.261434367	9.275520115	85.93129415	107.8692547	331.7241993	54800
’2003WU21’	0.908664875	0.544452232	28.53468955	57.58339781	140.6610161	207.2530838	54800
’2003WX25’	2.909766418	0.582826045	23.9677735	40.09193704	39.36702814	354.1681903	54800
’2003WY153’	2.46337963	0.59162191	1.177575708	210.9032736	211.3436506	106.1100698	54800
’2003XJ7’	1.24301797	0.465853347	18.17795088	254.0493936	271.6983879	174.6267573	54800
’2003XK’	2.338053777	0.713354503	2.329783126	246.0716038	104.6765198	155.8800901	54800
’2003YD45’	2.489246061	0.695643932	8.404940371	252.8516624	104.2994433	108.2628214	54800
’2003YG118’	2.282286598	0.64429509	8.129023147	348.579456	232.219348	116.2212074	54800
’2003YG136’	0.96887193	0.354891429	2.735200536	86.54730674	127.9875507	341.8604894	54800
’2003YH111’	1.419401335	0.486586942	4.367754156	91.8301615	84.16160466	297.9624153	54800
’2003YH136’	2.324326245	0.89383841	16.03711581	306.5501054	304.3438543	126.1468872	54800
’2003YL’	1.14607312	0.632458468	5.659853172	292.0590615	29.28201264	69.73794596	54800
’2003YM1’	2.605285429	0.522753806	13.65022898	291.4494317	223.7025673	38.52158072	54800
’2003YO1’	1.157611787	0.399017467	14.25725634	65.20496006	161.3898849	234.8782707	54800

'2003YP3'	1.34878359	0.493033586	37.28283421	260.4122766	282.1154004	12.60788238	54800
'2003YP94'	2.168156849	0.534221332	8.177483835	263.5380007	211.360627	190.6423859	54800
'2003YQ94'	2.653248595	0.618168544	8.531467754	272.5764867	85.68442189	87.57127099	54800
'2003YT70'	1.591150766	0.347533126	0.375769887	74.06500106	10.41540646	169.9015851	54800
'2003YT124'	2.31795811	0.47967808	5.45083487	327.1210488	117.4871815	148.786435	54800
'2003YW1'	1.6655605	0.297155583	19.41200909	277.0298055	182.3095794	104.7235939	54800
'2004BB103'	1.907117117	0.621804052	55.88798211	271.1937972	71.49405712	321.8105706	54800
'2004BF11'	1.892616884	0.406287127	1.806292471	277.1277301	199.0949719	313.287728	54800
'2004BG41'	2.513975126	0.611079974	2.954787602	301.7431204	152.9511632	83.21823697	54800
'2004BG121'	1.612418629	0.342853809	19.42971236	128.712282	206.6998061	266.1275119	54800
'2004BH11'	1.265868321	0.32788068	4.352311116	309.6989207	83.17478155	198.5796571	54800
'2004BH41'	1.194856641	0.500146474	30.45488184	123.2408498	200.3036903	30.6931992	54800
'2004BJ11'	1.751316879	0.397620023	4.268473218	269.1124864	112.1571015	86.95594198	54800
'2004BK11'	2.065154458	0.391817657	5.375922891	276.532918	211.7845439	225.8616729	54800
'2004BN41'	2.052050295	0.516920925	0.39909393	331.3247097	145.397124	235.7473638	54800
'2004BW1'	2.321240359	0.522070168	4.311830569	76.94629637	294.6505878	177.2262185	54800
'2004BY21'	2.422899274	0.473814112	6.543807163	40.42308518	155.5338334	72.85260193	54800
'2004BZ74'	3.048626295	0.891959667	16.60189783	233.8458705	121.3158853	336.4716028	54800
'2004CE39'	1.027225291	0.391328423	17.70677067	153.2987871	126.0474412	127.6117085	54800
'2004CO49'	1.375401923	0.388564895	4.261901493	115.2391077	315.8051705	25.78486877	54800
'2004CZ1'	1.539659326	0.45454907	1.998748716	146.8986314	65.75981307	156.0454042	54800
'2004DH2'	0.94404479	0.400224631	23.02316035	157.3498478	216.0754526	180.7595412	54800
'2004EK1'	1.251061358	0.251493124	11.63288811	167.0229756	300.3932326	175.4379027	54800
'2004EN20'	1.865839293	0.379365823	40.12491088	185.6259166	246.3902062	2.859144982	54800
'2004FA'	2.122931061	0.590174778	9.193611852	180.208835	55.90306581	173.2626084	54800
'2004FD'	1.301702933	0.674241711	1.020461995	49.90302006	357.1829859	102.8504833	54800
'2004FE4'	1.390709189	0.234984684	15.76135209	358.7335469	198.8177877	300.0265046	54800
'2004FG11'	1.589031592	0.724111133	3.109512451	84.74332408	227.4223695	92.36812521	54800
'2004FP4'	1.997750119	0.469333692	2.122677391	24.67040288	196.8526068	225.1233328	54800
'2004FU64'	1.837339483	0.367176623	24.87792891	20.98297329	286.3460411	249.8098009	54800
'2004FW1'	1.602201084	0.714536351	10.12655311	195.0018863	90.79566095	87.47309013	54800
'2004FZ1'	1.79388837	0.530981858	52.62938861	152.6756735	89.56669847	311.9446683	54800
'2004GD'	1.064414292	0.307586136	6.222098651	26.72126523	281.0066804	9.742908817	54800
'2004GD2'	2.035052999	0.502728709	1.775972581	44.7550659	144.0999181	218.2446302	54800
'2004HC'	0.789175779	0.598753449	28.97489489	203.022814	159.3271279	92.69855689	54800
'2004HD2'	2.313100707	0.833536892	15.40668804	26.78987451	61.88382111	125.9715643	54800
'2004HE'	1.773483526	0.60843577	9.479418568	208.3025211	79.36530926	323.7965503	54800
'2004HO1'	2.206477691	0.521505041	25.76635853	43.54090026	265.0538995	113.426382	54800
'2004HT59'	0.97996524	0.223414346	11.13470324	214.7074081	112.1090386	180.0821926	54800
'2004HW53'	2.178397664	0.581166869	39.06766926	32.9035381	95.68817442	174.3114784	54800
'2004JA'	1.355766614	0.628413626	29.61572032	56.98928314	54.25960438	359.2381118	54800
'2004JB12'	2.188413082	0.517131128	8.43513676	152.0850977	152.2046871	120.0802367	54800
'2004JG6'	0.635140312	0.531242432	18.94546979	37.0606571	352.9650734	205.4283627	54800
'2004JN1'	1.085333447	0.175585891	1.496836843	144.0413373	1.922803439	73.36779102	54800
'2004JN2'	1.067680532	0.608375576	19.76087986	228.3838663	179.8733158	224.6178735	54800
'2004JO20'	1.469026111	0.432714431	10.25417695	234.5116418	68.08380005	170.5601296	54800
'2004JP12'	1.860959099	0.771364087	8.163070858	216.9879195	262.6902565	304.7580673	54800
'2004JW20'	0.952773881	0.561536675	14.73070241	235.240851	207.4536364	58.24600172	54800
'2004KZ'	1.289416872	0.374778818	12.4104022	58.46971899	100.9215168	74.91234114	54800
'2004KZ14'	1.536982418	0.312983829	7.88514251	200.3276053	310.4905023	190.8164171	54800
'2004LC2'	1.88051217	0.736948617	11.00495742	84.6161849	290.3500481	242.2184042	54800
'2004LK'	2.090303409	0.512592219	7.814525366	76.10216511	229.8458235	158.2125706	54800
'2004LX5'	1.304019067	0.211803292	13.57737888	83.30700686	172.857088	3.271434673	54800

’2004MC’	2.442012877	0.587859478	2.435752494	91.5060602	203.254745	53.36746243	54800
’2004ME6’	2.366275151	0.575394677	9.437839011	112.2016376	210.3431028	64.46246545	54800
’2004MP7’	2.74547793	0.716147963	17.21498855	97.95421662	109.5335419	359.9971764	54800
’2004MQ1’	2.405400018	0.70672077	10.96292213	57.64855759	330.9909716	45.91208589	54800
’2004MS1’	2.274077847	0.590200591	6.967462128	263.3315245	324.2849916	116.2228101	54800
’2004MW2’	1.145054698	0.638066017	35.11611123	96.20115838	50.11471596	275.9595181	54800
’2004NC9’	2.694453093	0.556430774	23.92896743	335.3346585	259.2694041	25.09124866	54800
’2004NF3’	2.063467586	0.479689983	0.785849351	270.135767	22.61392927	172.0156462	54800
’2004NK8’	1.326589304	0.280805313	15.93346042	290.0270837	302.6132508	344.9938788	54800
’2004NL8’	2.566768848	0.72091849	4.973408523	157.6725735	270.881908	356.8391232	54800
’2004OF6’	2.187860091	0.413545951	8.953323575	39.14561726	262.7710616	124.9191694	54800
’2004PB97’	1.234845782	0.16278859	10.96756596	140.7427421	160.4862588	62.3522591	54800
’2004PJ2’	1.417950709	0.341943135	2.583396562	317.234927	281.5994297	238.9247197	54800
’2004PR92’	1.133223756	0.340476678	13.00528104	321.5963133	100.2542035	141.1638651	54800
’2004QD20’	2.166789652	0.701387782	17.56504474	224.3432665	353.1087066	152.6058741	54800
’2004QG20’	2.341672601	0.449745811	7.634459815	169.6288218	208.5108642	57.87915031	54800
’2004QJ13’	2.028088213	0.481341649	3.022970176	147.5218997	185.8377691	171.5639172	54800
’2004QX2’	1.286534212	0.902644361	19.06130604	320.325423	218.6574526	9.417768731	54800
’2004RB11’	2.111884726	0.506254283	3.201750732	161.7496874	194.0199456	133.6109005	54800
’2004RC11’	1.82573774	0.466477239	1.859458243	147.67136	161.9219363	269.8250863	54800
’2004RC252’	1.306334985	0.129078741	19.32548346	175.4075754	56.63965747	43.0225854	54800
’2004RD84’	1.820476098	0.415922331	24.24380585	331.4078449	100.2092439	210.9620975	54800
’2004RK’	1.388646741	0.300406733	18.14661573	178.8550346	264.3242031	143.2738252	54800
’2004RK9’	1.838371031	0.426054605	6.226446881	355.0846096	283.9154082	277.4026234	54800
’2004RN111’	1.668137257	0.393659597	12.60676449	349.8209804	331.3391727	355.8570235	54800
’2004RQ10’	1.86458451	0.44175834	5.687206939	335.3141046	349.4229677	244.5670799	54800
’2004RQ252’	1.126055372	0.388545804	7.743883942	25.55918324	82.91150538	117.4903027	54800
’2004RS109’	2.331103231	0.495625776	33.95789932	173.2773834	192.7592555	61.98698774	54800
’2004RU109’	1.532276665	0.4891129	5.848367043	171.4280266	250.5643034	53.84786648	54800
’2004RU164’	3.368038174	0.617786364	12.70187051	114.3949475	232.6444647	244.9322067	54800
’2004RU331’	1.241371957	0.244038129	17.46394947	201.6472203	261.7526299	296.2432661	54800
’2004RY10’	1.672739709	0.621213376	15.47550396	165.9115306	283.2670321	310.6719741	54800
’2004SA’	2.263542032	0.537276879	18.4014051	173.8851663	183.2410273	83.92016646	54800
’2004SA1’	1.185011003	0.150444139	17.11802944	355.9313496	327.7597558	116.0304073	54800
’2004SA20’	2.413935965	0.71126999	2.848238704	135.6314685	146.9779987	54.51461138	54800
’2004SB20’	1.182950682	0.413124557	30.27890718	30.95515064	209.3325487	175.310224	54800
’2004SB56’	0.865729586	0.237942315	18.69681569	302.11	233.4441192	262.0750681	54800
’2004SD20’	0.875078561	0.464942424	21.33378893	46.64092042	94.3832741	296.6177296	54800
’2004SD26’	2.036256536	0.775649469	4.872761947	117.7244046	359.3696994	130.3214215	54800
’2004SS’	2.195222882	0.526968642	6.515940972	1.509473678	352.8734929	106.143233	54800
’2004ST9’	2.247953008	0.435350196	12.25170005	195.2720724	59.66890116	161.2110511	54800
’2004TB10’	1.103143233	0.09356588	22.49960893	12.8937274	149.7077464	63.34267323	54800
’2004TK10’	1.058062787	0.298931309	24.59773273	205.354915	347.6628162	126.6169989	54800
’2004TL10’	2.662420583	0.652515647	9.12518127	12.16847478	322.6757045	351.4365583	54800
’2004TL19’	2.519644628	0.534431843	12.62208595	209.8887955	194.5783576	4.51674983	54800
’2004TN’	1.428172834	0.435693289	14.04205336	17.20599222	159.8493145	25.5242172	54800
’2004TN1’	2.749642226	0.697540942	8.442669871	214.0454836	233.4155046	310.1301203	54800
’2004TO20’	1.439032406	0.295572196	9.555979199	207.8211807	123.5911914	168.338576	54800
’2004TP1’	1.290560271	0.389282961	7.484580741	30.15506808	87.66441307	243.9157737	54800
’2004TP20’	1.343911947	0.350874164	25.37209832	213.588012	264.830693	181.597908	54800
’2004TR13’	2.018190979	0.729490647	17.86543805	12.38295699	249.7839597	180.0128618	54800
’2004TT12’	2.498907912	0.545084928	0.722697121	184.0341334	197.6205244	17.36509152	54800
’2004UL’	1.266465535	0.926748377	23.70376592	39.71768181	149.4228671	268.0173732	54800

’2004UT1’	0.964338887	0.221087422	4.507340384	211.9705677	294.2279247	28.42363163	54800
’2004UU1’	1.226277969	0.273640424	29.96006223	217.8228246	113.5414548	44.97422897	54800
’2004VA15’	2.884486922	0.596467146	17.41870934	36.06489732	330.1000245	303.1208755	54800
’2004VB61’	2.300379309	0.495101114	9.249792382	59.84723677	344.0965043	58.2100078	54800
’2004VH1’	1.55048975	0.471555407	32.76991543	43.38221135	88.60124248	1.409392341	54800
’2004VP’	1.65483419	0.429379661	11.7006036	45.87693733	70.93448521	296.5290651	54800
’2004XB45’	1.550351937	0.581298591	3.161816152	85.03651688	85.99107381	352.2457906	54800
’2004XD6’	1.948231764	0.564737856	6.676912476	66.80847111	88.39243667	143.5770239	54800
’2004XD50’	1.825200045	0.377157988	20.66079938	106.1654375	63.64040751	172.6507261	54800
’2004XG’	0.837385169	0.298282642	1.203504747	285.1787603	0.956198972	193.9440883	54800
’2004XH29’	1.382203977	0.503519775	22.60817996	232.1348482	334.016831	94.69610843	54800
’2004XJ35’	2.116322487	0.390469762	7.649570775	72.89092107	7.594669002	107.1441213	54800
’2004XK35’	1.952497829	0.41634174	31.48384203	263.9528775	64.0071483	222.9210029	54800
’2004XM130’	2.324536316	0.46587958	28.22156934	309.3204383	189.779422	39.84026631	54800
’2004XN29’	2.425511965	0.705515586	2.381346476	320.0501815	36.94417314	31.28517585	54800
’2004XN44’	2.422243561	0.579744824	2.765594969	178.8543574	83.01267982	49.75884004	54800
’2004XY60’	0.640243133	0.796766108	23.75213585	122.6781581	130.7915191	122.7197734	54800
’2004YC’	0.868335094	0.313304986	6.067147464	263.470442	47.30289962	62.0512695	54800
’2004YG1’	1.009367532	0.158530813	19.82754187	271.8385767	293.9620408	226.7895742	54800
’2004YZ23’	3.423849458	0.676827223	56.11498537	253.0550987	300.9407172	207.5787653	54800
’2005AD3’	2.408107798	0.500084338	14.5532561	294.5947465	277.1541251	330.4151092	54800
’2005AU3’	1.246427499	0.473639168	3.779873111	105.13211	266.7345759	329.2176083	54800
’2005BC’	1.189770163	0.278016714	30.12452051	292.5121823	84.17738335	38.04835292	54800
’2005BE’	0.883822267	0.421171221	31.18904673	116.0008928	168.6771598	60.13756025	54800
’2005BG14’	1.992903286	0.727995885	21.64623144	95.43906935	280.4758742	153.7583621	54800
’2005BG28’	1.025771871	0.227092168	6.132406589	313.5247276	80.85090638	317.5971831	54800
’2005BN1’	1.785860191	0.554118506	6.777651048	300.6352869	107.7586099	243.4351855	54800
’2005CA7’	1.830102145	0.441043666	6.228983031	316.1932888	176.8235139	196.5117802	54800
’2005CD69’	1.088423593	0.187595426	2.781808096	336.6049239	264.4538692	47.28409755	54800
’2005CF41’	1.648841214	0.586051311	15.89382731	132.2790377	208.6072445	57.53827769	54800
’2005CN’	1.015752478	0.184990673	2.313581729	308.7588122	321.1803878	145.2206292	54800
’2005CS6’	2.618920784	0.547435305	23.51802358	314.9058545	267.9161884	296.3301664	54800
’2005CW25’	1.625719985	0.477907425	28.47375402	148.4339683	108.8305562	258.5487209	54800
’2005EB30’	2.16680793	0.498956934	6.858946889	167.9431972	22.40942258	55.62550352	54800
’2005ED318’	1.847480384	0.448403603	2.391849319	82.13989582	164.0652433	143.6453126	54800
’2005EH94’	1.216945268	0.26732903	6.193878098	348.8666342	255.0082404	230.0804022	54800
’2005EL70’	2.276089346	0.92476618	15.95865558	166.9192965	221.0996494	43.26590739	54800
’2005EN70’	1.778878087	0.366687591	46.14391428	175.7144001	10.10446278	201.9194654	54800
’2005EO30’	1.225144711	0.160750382	14.00963523	347.699457	191.8473461	264.0150077	54800
’2005EQ70’	2.4014516	0.474587015	6.222130641	70.76629671	102.4501778	1.73371485	54800
’2005ES1’	1.354988842	0.294710531	1.85464365	164.646726	315.2560826	158.0528487	54800
’2005ET70’	2.129150044	0.828390185	5.577231106	82.96121638	326.8777909	88.23041352	54800
’2005ET95’	1.862073231	0.417166865	24.78479806	357.4256894	104.5000458	200.6271015	54800
’2005EU2’	1.5079546	0.353231869	4.577802087	191.1329503	23.9704955	341.2399689	54800
’2005EY169’	1.30801027	0.233072389	20.89000968	346.9804987	227.5079072	145.5070713	54800
’2005EZ’	2.337326947	0.48732098	10.4956383	215.4736587	286.6574172	21.99394764	54800
’2005EZ169’	1.316015467	0.214921822	2.742064345	175.9042429	353.4710434	168.0079481	54800
’2005FH’	2.696848746	0.656670546	34.84028198	144.156861	317.8869719	322.0491257	54800
’2005FN’	0.933047105	0.330234645	3.748189396	177.4138347	120.858591	314.999464	54800
’2005GA120’	1.286746858	0.386443133	12.26139332	197.728335	282.6622159	217.6183499	54800
’2005GC120’	1.192970386	0.497515276	16.537155	68.05220616	258.1157342	209.5400639	54800
’2005GC141’	1.489665664	0.198852105	28.2797899	186.0088591	40.52335435	350.7138896	54800
’2005GE59’	2.109477325	0.601323692	16.10034754	195.0100286	242.7800137	130.9125236	54800

’2005GG’	2.040933471	0.659868545	34.80381081	106.4841771	335.2349215	106.1157844	54800
’2005GH’	2.174805418	0.478869533	20.20662298	189.7742327	306.0350414	72.16612278	54800
’2005GJ8’	1.767064046	0.550886411	3.399379193	252.8319426	214.450921	228.4433262	54800
’2005GL1’	2.508871786	0.515317026	3.214752375	189.4189484	19.5714189	326.6626686	54800
’2005GM162’	2.245268266	0.494338821	1.975803973	3.552073778	257.6728754	10.21967046	54800
’2005GQ33’	2.339179053	0.731503423	1.55316151	73.62754183	36.47980697	21.10185741	54800
’2005GU’	1.551463448	0.614816284	25.0464974	27.05566744	275.6371308	285.1329221	54800
’2005GW119’	1.640727343	0.233187174	2.88082599	171.1416555	242.8115397	43.7059078	54800
’2005GY8’	2.048361521	0.67234766	2.825886347	179.9590165	103.2034556	62.77031338	54800
’2005HB’	2.702076822	0.607078829	9.215964403	96.75150035	125.3097928	290.5638727	54800
’2005HB4’	1.354808802	0.228133051	2.527210888	83.86106916	153.1999723	87.28396453	54800
’2005HD4’	1.440484547	0.262357588	22.47087344	224.7605269	349.5999915	26.3902176	54800
’2005HM3’	1.678365892	0.309922097	28.31914126	209.2738858	4.107080938	238.4252025	54800
’2005JD46’	2.648037762	0.784752031	19.03380975	49.1326665	49.82326703	312.5128116	54800
’2005JE46’	1.904142343	0.552408963	8.258963363	238.5951628	114.4603346	73.89199304	54800
’2005JF46’	2.69519803	0.5352167	35.98800889	80.54245077	197.3873272	276.4314954	54800
’2005JF108’	1.947648192	0.792870879	34.103474	69.22294564	23.91686414	146.9862019	54800
’2005JJ91’	2.793635518	0.596803333	24.4377053	67.04427706	206.1177068	265.2892598	54800
’2005JN3’	2.151971928	0.506323264	9.183457388	205.9941046	49.01832004	31.83552834	54800
’2005JR5’	1.159540136	0.154671199	31.10020596	49.35437957	246.5515116	255.036665	54800
’2005JU1’	2.340825669	0.684160966	6.077911928	35.63869808	266.5666157	344.1580506	54800
’2005JU108’	2.124794149	0.461690997	6.532164901	188.1060625	106.0659666	34.41634888	54800
’2005JV1’	1.912023366	0.465562998	3.431957631	217.0843286	356.2663226	130.1617925	54800
’2005KD7’	1.166932371	0.555340185	40.05359991	240.1289082	137.4960595	214.7009721	54800
’2005KR’	1.138903118	0.3059954	21.63033323	63.40406441	320.7370107	203.9093205	54800
’2005LP40’	1.963897552	0.546965095	23.60795759	148.7077386	186.5449849	71.33245044	54800
’2005LU3’	1.057035107	0.308351884	5.580244656	80.77648888	71.77897997	141.3401706	54800
’2005LV3’	2.208876216	0.438695	7.306736315	250.8820341	13.51259201	22.85823397	54800
’2005LY19’	1.60182658	0.240114978	30.00412613	338.4803139	120.1811245	62.4554587	54800
’2005MC’	2.616759751	0.593072635	27.28066142	287.4314494	125.0123628	281.0438205	54800
’2005MG5’	2.145673724	0.458244941	6.714133752	261.2329581	46.28810165	20.13264938	54800
’2005ML13’	1.148007716	0.246261617	6.83641982	140.7869716	220.6075759	222.9795422	54800
’2005MO13’	0.863515305	0.410979372	6.317590806	176.7269629	250.1256549	327.7539206	54800
’2005MP13’	2.150540491	0.451480874	7.315943087	114.8908844	182.3499246	24.96558617	54800
’2005MW9’	3.584529497	0.887052439	55.30155234	291.8080986	241.5107269	192.1699419	54800
’2005NE21’	0.789296649	0.49638216	10.63894348	289.8269391	194.6332905	96.23331753	54800
’2005NG’	1.783030526	0.505731783	8.457820868	276.2099525	305.9093096	174.2659638	54800
’2005NG56’	2.80387981	0.649025161	16.72676459	114.5276737	150.3726466	263.7063726	54800
’2005NJ63’	0.86927114	0.422569141	26.589658	120.8931073	1.695411366	230.2975036	54800
’2005NW44’	0.779312426	0.483396222	6.056461599	114.5572299	0.648752381	155.2946321	54800
’2005NX44’	2.215227452	0.905855299	37.23150694	309.6793979	214.5351565	29.36157725	54800
’2005NX55’	1.52294619	0.587566081	26.17147547	106.3862611	277.2834499	255.9199489	54800
’2005NZ6’	1.833509482	0.864613757	8.504513139	39.62926132	48.09154086	96.32667797	54800
’2005OF3’	2.383591248	0.588189101	3.282271387	174.2986194	94.66032303	335.0071349	54800
’2005OJ3’	2.708964562	0.537753487	4.441970728	239.0331571	155.0169999	238.6926721	54800
’2005OU2’	1.234831917	0.373662915	47.77706582	127.388817	310.1642771	78.37373265	54800
’2005OW’	2.666087554	0.601567804	1.639149244	271.7814339	62.24531914	270.4426422	54800
’2005PO’	1.252199533	0.373094784	12.51816956	300.615136	249.3957568	227.0472696	54800
’2005PY16’	1.975868043	0.524587696	6.4153256	159.426775	193.3781723	68.91350356	54800
’2005QB5’	1.122565486	0.252011049	13.67908686	150.0279169	93.62235169	329.2839688	54800
’2005QQ30’	1.754820646	0.5451692	11.2290824	67.95441817	173.530536	175.0276089	54800
’2005QR87’	2.135946823	0.426488807	8.616190731	354.1391461	345.9311613	16.99023472	54800

’2005QX151’	2.296878012	0.472824305	7.974924276	359.973229	36.28826112	318.827027	54800
’2005RC’	2.150401917	0.752939481	16.33545067	192.2432697	40.31751917	35.30076573	54800
’2005RD34’	1.326417314	0.264614807	19.919738	175.6026883	102.0032352	91.88039095	54800
’2005RN33’	1.733566606	0.256527107	7.190185311	100.7562033	300.0586891	124.4497007	54800
’2005RR6’	2.969336487	0.697897249	6.966466283	28.6673223	58.62914914	220.9977284	54800
’2005SC’	2.271382704	0.663522436	6.613947664	281.1961132	156.5071924	319.37691	54800
’2005SG26’	2.476597881	0.592412451	5.904627846	184.8982201	207.8710415	287.4353419	54800
’2005SH19’	2.269393635	0.857968653	47.64097505	18.59280419	158.2046019	282.0599955	54800
’2005SN25’	1.229902517	0.270441186	13.00958914	168.3839696	45.06086921	250.7635055	54800
’2005SO1’	2.168532963	0.577016081	5.234725911	358.9656636	315.4877058	9.978147514	54800
’2005SP1’	2.37580827	0.668293865	5.983466581	180.1230404	109.7287768	326.0708697	54800
’2005SR1’	2.221846929	0.497465549	3.24188638	147.8565955	135.0513047	20.34619897	54800
’2005SS4’	1.460089538	0.751245502	14.59314075	28.73142171	192.1607528	343.652486	54800
’2005SV4’	2.395369798	0.596114731	7.96702675	174.5814913	140.0906096	318.9388223	54800
’2005SX4’	2.727251144	0.574986138	3.536349355	171.1946329	250.1186505	242.4177625	54800
’2005SY70’	2.28007754	0.53912002	1.438007742	19.06113315	358.5709861	328.7935946	54800
’2005TC’	3.726920038	0.721586682	14.9797336	9.027963497	322.0780145	162.6819173	54800
’2005TD49’	2.686448275	0.622637926	0.09325682	196.5002723	191.7015092	254.9613528	54800
’2005TE’	1.755282837	0.579457712	6.514189926	13.04512496	270.6185665	155.7445431	54800
’2005TF45’	1.156047688	0.073985016	6.806169374	197.196284	37.29282955	329.3744275	54800
’2005TF49’	1.041935363	0.025438959	24.54991821	25.76238079	302.3578537	48.65725156	54800
’2005TM’	0.84133278	0.416435895	5.202441075	8.517201496	151.9661246	272.9242158	54800
’2005TN’	1.765844493	0.301106054	16.85119175	13.36395319	313.2072012	147.2250543	54800
’2005TP’	2.280207672	0.587145899	7.868824698	86.46845479	353.679905	312.0743898	54800
’2005TR15’	2.116789907	0.432136353	3.918100052	214.2682435	59.08889651	68.01095846	54800
’2005TU45’	1.9742637	0.495648561	28.54507146	120.2630466	76.89448754	82.42524377	54800
’2005UB’	1.895572788	0.738643568	27.72199835	272.1626121	3.087655243	88.19781227	54800
’2005UF’	1.674991008	0.515419137	7.282976177	209.9719735	244.3179185	135.0175388	54800
’2005UF1’	2.185232075	0.431509762	13.5785393	49.57884008	1.259435249	337.7771025	54800
’2005UH’	1.626011025	0.380992261	6.32436937	36.48635769	307.5194684	200.6871605	54800
’2005UH6’	1.000664063	0.632304979	2.648426389	19.20915123	200.2531526	106.7285829	54800
’2005UL1’	1.367480019	0.230806685	18.38268913	35.74025997	37.81188771	313.4397717	54800
’2005UN157’	2.546242465	0.855333336	44.83986287	21.79819397	209.9906244	312.6011879	54800
’2005UP64’	2.689310513	0.579281214	23.69818063	202.7061752	280.4290885	220.1377308	54800
’2005UQ’	2.244178516	0.547039179	8.709943801	25.59291824	336.8489891	339.7038352	54800
’2005UT64’	1.821332548	0.522813089	10.48907144	161.7162158	327.136411	57.46585681	54800
’2005UU6’	2.478509621	0.501759931	16.82881407	212.6972318	190.0797556	282.9166675	54800
’2005UW5’	1.397354179	0.395133406	2.940546106	35.20344515	63.0514972	284.8940489	54800
’2005UY5’	2.227320001	0.416993958	7.136187464	56.66409864	261.8681709	27.59243716	54800
’2005VB7’	1.801915506	0.347563842	32.58234924	241.5194104	212.2904871	69.23831557	54800
’2005VE’	1.063672325	0.275200614	22.41659306	223.4278133	31.69315547	63.44871235	54800
’2005VJ1’	2.639034136	0.526602096	11.27494088	330.9907569	76.84161041	253.3955548	54800
’2005VO’	2.438792896	0.671165136	0.48224813	204.2987972	254.5240958	277.8506118	54800
’2005VR7’	1.08285778	0.287173463	25.24850524	254.9224722	101.0288498	309.8608468	54800
’2005VT2’	2.324369675	0.565013896	6.14329564	200.5126122	155.0358766	322.3445378	54800
’2005VY1’	1.673342099	0.410943236	6.607822777	45.08614926	18.15757882	142.109382	54800
’2005WE55’	1.742270981	0.397835276	23.53454329	63.30895029	87.31290763	67.84522448	54800
’2005WF4’	1.509701864	0.232912883	15.96061911	58.41974335	21.57292514	211.5249833	54800
’2005WG4’	2.437487616	0.601910305	10.34520886	63.65499322	309.2688574	296.2455486	54800
’2005WK4’	1.01165768	0.237172209	9.832971868	138.3079943	74.34563918	218.5373502	54800
’2005WM3’	2.671936097	0.621299073	1.230880521	240.4472043	190.0938311	247.774205	54800
’2005WO3’	1.573600146	0.354902896	22.95368963	64.20302095	312.4808455	212.1926856	54800
’2005WR2’	1.531501317	0.546277729	7.858207759	282.475663	49.43230205	248.5414802	54800

’2005WR3’	1.297014792	0.294100249	34.24815347	65.96500459	265.7018104	74.67148786	54800
’2005WS3’	0.671710111	0.575098558	23.02970098	69.43286353	176.050417	1.572083636	54800
’2005XA’	2.554217603	0.656231272	5.499706456	69.51000457	45.40757104	254.9773294	54800
’2005XC1’	2.519593249	0.589037749	12.9702906	79.18793436	349.8986404	270.3323416	54800
’2005XL80’	1.726097361	0.487530668	10.89334507	53.45981227	143.4534545	63.90394874	54800
’2005XM4’	1.303779973	0.06414282	34.14484148	81.91410849	345.2140955	358.0872983	54800
’2005XN’	1.7558843	0.417028719	21.13818041	71.87134793	336.047859	111.9026556	54800
’2005XN27’	2.405402125	0.633458558	0.295596616	215.5373553	169.4889819	297.6203872	54800
’2005XP66’	2.804118866	0.580182935	2.904008097	215.881942	273.183402	214.2576729	54800
’2005XW77’	1.615358615	0.355587986	16.44089626	268.8342719	99.99230186	200.5571459	54800
’2005XY4’	1.056246143	0.598528615	1.907278501	163.1541786	143.3156032	324.1101538	54800
’2005YC’	3.233825465	0.612156674	24.21097433	296.5309478	123.4983597	188.7083925	54800
’2005YK’	1.061114662	0.307674427	5.621922941	269.6812915	80.38301272	312.7830135	54800
’2005YN128’	1.66153217	0.442624114	2.578997109	284.9098802	117.7381248	150.6720463	54800
’2005YP55’	2.282844984	0.43685745	6.934734066	105.344423	267.5274349	343.7534299	54800
’2005YP180’	1.37297091	0.617083183	4.112925807	289.1909172	92.15675594	321.3376044	54800
’2005YT55’	2.251623991	0.460222617	2.300131436	141.397384	11.63415939	294.0471121	54800
’2005YT128’	2.614667925	0.571442394	27.23314058	294.1404287	115.2622929	262.2347172	54800
’2005YU8’	2.012120997	0.542691717	4.031543719	279.2588839	237.3328056	352.2980587	54800
’2005YY36’	1.893321836	0.39558266	4.614268842	275.2987903	213.9906796	37.94148138	54800
’2006AC3’	2.337417333	0.567849547	3.059553277	108.5341082	323.1884935	299.7018283	54800
’2006AD’	1.048341233	0.489887754	54.98707887	120.4090217	87.3052664	166.4766625	54800
’2006AH4’	1.917360219	0.470667966	0.663858332	279.1960296	208.8028407	25.93168508	54800
’2006AL4’	2.491428298	0.585285015	2.761403374	103.1664279	27.97436534	259.9521996	54800
’2006AN’	1.093475388	0.219971207	7.40400593	277.7204825	273.4371125	130.447428	54800
’2006AO4’	2.629712858	0.582417898	24.40164703	318.6284686	50.31731064	315.2742379	54800
’2006AP3’	1.722149207	0.373532502	10.00705421	289.8035012	172.2430432	103.5437324	54800
’2006AT2’	2.709321798	0.598210331	21.15584761	144.1663721	39.19241375	204.9670041	54800
’2006AU3’	2.266766939	0.565231969	1.416723181	114.7072183	0.051167672	303.2098687	54800
’2006AX’	1.290200226	0.141918408	11.68189996	280.5765243	232.7595466	316.8685431	54800
’2006BB9’	2.692871886	0.666886172	6.744058464	301.5732089	133.2615101	240.8347628	54800
’2006BC8’	1.227503583	0.431873768	6.901676483	303.4039899	91.10816964	76.42542756	54800
’2006BE55’	1.196243223	0.424688099	2.423703331	125.744764	125.8644007	354.6203721	54800
’2006BG’	1.682327223	0.732371163	4.813774902	281.1111813	341.0293493	70.44594499	54800
’2006BO7’	1.435195297	0.403149677	0.342534742	295.2219008	247.508062	211.3004048	54800
’2006BP147’	1.287136729	0.240379367	5.596261112	310.7831892	123.1535268	14.02578024	54800
’2006BT7’	1.521988193	0.63309142	16.14750798	298.6737779	342.4252015	64.70698748	54800
’2006BX147’	0.82439941	0.67311291	9.885953	143.270844	202.830405	358.134321	54800
’2006BY7’	1.873876851	0.455637228	3.178048423	302.7456652	162.3374837	46.43146401	54800
’2006BY8’	2.485913623	0.673220983	2.688478299	300.7713951	122.4230683	272.4290744	54800
’2006BZ147’	1.023388406	0.098644885	1.408921578	139.8459247	94.70137815	171.0524632	54800
’2006CL’	1.662574422	0.412952402	8.147057197	312.9841207	151.6742138	125.7784032	54800
’2006CN10’	2.800263864	0.607456386	1.236877858	229.4848284	312.2705296	207.3909099	54800
’2006CT10’	2.89869761	0.660174283	12.93769569	111.483196	30.01049685	204.9975133	54800
’2006CU10’	1.475186212	0.447464826	49.23682776	146.3228101	130.6421567	118.3802782	54800
’2006CV9’	2.729665528	0.527295725	19.5694015	116.5839367	32.98020978	224.8154102	54800
’2006CW’	2.227632201	0.492111282	6.542805332	144.8803116	41.31720155	289.7837422	54800
’2006CX’	1.714030766	0.296109848	28.9537403	354.8672333	160.7904756	83.89131446	54800
’2006CX10’	2.554580446	0.517606176	27.33369346	32.42417221	188.9746925	230.6016253	54800
’2006DM’	1.355156736	0.212117841	7.074477068	333.9955518	177.6941907	272.7521665	54800
’2006DN’	1.380099766	0.275807843	0.266066675	96.33940368	101.420943	229.9281648	54800
’2006DP11’	1.859926238	0.697830657	8.544994492	21.51091222	31.00999776	58.50583151	54800
’2006DQ62’	2.029453024	0.682547645	5.767906	217.9455648	6.821389718	328.2065619	54800

’2006DS62’	2.263084132	0.569097301	1.884239348	153.4418114	38.2297902	283.8910452	54800
’2006DT14’	1.91101633	0.566673292	8.655118964	78.29281604	178.2140521	348.2784729	54800
’2006DT63’	2.384313175	0.47187039	4.715187748	335.8896154	188.9842714	269.1509563	54800
’2006EA’	3.168080808	0.629528678	24.19797596	1.444167892	149.9182748	174.8547449	54800
’2006EC’	1.137563865	0.217519576	7.578164874	347.6112412	245.5227792	46.0786724	54800
’2006EF1’	1.262749806	0.342219233	13.60358649	328.2371265	296.4836933	272.7800841	54800
’2006FJ’	1.179261588	0.287534556	27.71582821	187.7037103	229.1382087	142.9564368	54800
’2006FJ9’	1.748945697	0.345343684	4.122382242	166.5425144	101.4516739	8.482544905	54800
’2006FX’	1.49539657	0.438423382	24.62157795	181.2627082	299.3892194	196.0912982	54800
’2006FY35’	1.812872436	0.359468589	21.24982775	26.46303695	207.2488348	12.65360977	54800
’2006GA’	2.457058622	0.616458575	18.01422618	188.2869267	304.8475656	260.1604912	54800
’2006GB’	0.958987887	0.179390487	10.06147021	183.8515764	242.8419633	40.78659446	54800
’2006GC1’	1.704841953	0.816347037	5.983563097	196.4135701	232.5854355	98.9791255	54800
’2006GQ2’	1.29063607	0.465741927	25.84115206	13.96936186	64.48934005	344.898337	54800
’2006GT3’	1.72144492	0.36071067	8.669528284	11.06671688	223.6326253	41.92682526	54800
’2006GU’	2.703409759	0.581007982	17.56169338	149.4575493	145.7843033	174.9080617	54800
’2006HA6’	3.250328652	0.633724837	22.70759044	17.00386421	157.0665956	167.7402291	54800
’2006HF6’	1.406264446	0.549845165	6.583676343	29.53651697	86.38428625	237.1595173	54800
’2006HH56’	1.364092427	0.324370407	23.83620198	40.6155216	238.434451	193.5144063	54800
’2006HR29’	0.985193843	0.263331202	9.545781796	232.7674902	212.5625604	332.4728987	54800
’2006HS30’	2.360474913	0.57194705	2.301468461	208.3728535	20.6363993	255.0100974	54800
’2006HT30’	2.568988714	0.616587993	1.641125995	73.64236119	224.096325	202.4452931	54800
’2006HU30’	1.454915943	0.41996227	24.00310831	44.73445043	274.1907349	109.5397722	54800
’2006HW5’	2.394094884	0.568934408	6.017576062	26.7553833	176.5321261	254.9589143	54800
’2006HW57’	2.144668488	0.520561935	7.152652351	251.2355473	33.55593103	274.1878471	54800
’2006HY50’	2.585270774	0.628344342	25.68377607	44.67871109	77.67636455	243.5551757	54800
’2006HY51’	2.602038462	0.969050864	30.5253584	42.428821	340.4874386	206.0783468	54800
’2006HZ51’	1.897757049	0.449535744	12.41161842	84.35326432	193.2254394	320.2359229	54800
’2006JE42’	2.639770527	0.591398473	5.372850581	350.3448158	263.2181863	210.9013032	54800
’2006JF’	1.084610255	0.658040061	42.60233121	216.2917796	211.0327535	188.8182175	54800
’2006JT’	2.400537184	0.485828311	36.46394076	21.05480395	161.9391334	255.0686642	54800
’2006JX25’	2.151341961	0.432433953	3.077716968	44.31258737	262.2376495	254.7085181	54800
’2006KA’	1.63338032	0.561561436	31.03019321	236.1585644	244.5457683	116.3529761	54800
’2006KC40’	1.667974025	0.529286537	8.815561872	78.07918463	260.2255232	26.1286778	54800
’2006KF89’	1.87010875	0.309135169	38.17973405	246.3191416	72.58119584	313.9806035	54800
’2006KL89’	2.74048957	0.547661181	13.7281233	84.32585932	155.1188647	203.723752	54800
’2006KQ1’	1.244262686	0.175419545	9.604364894	88.72710239	18.1718882	50.01946703	54800
’2006KS38’	1.431453318	0.456231417	28.71681862	63.47735164	101.1119986	201.5019783	54800
’2006KT1’	2.305919342	0.474504456	9.266984338	138.7576417	188.8540808	210.194579	54800
’2006KV89’	1.150231097	0.272803648	3.554754405	71.81409695	87.70068611	64.37564785	54800
’2006KY67’	2.07627429	0.535747568	3.025246983	117.2754488	173.854859	289.7502976	54800
’2006KZ39’	0.609413238	0.541159457	9.928110188	41.48167136	354.3277802	296.3122053	54800
’2006KZ112’	2.524171205	0.886899429	37.76601985	166.2672409	358.143727	239.3416911	54800
’2006LC’	1.48141477	0.357404116	11.73502411	249.7001979	52.25487934	113.3936662	54800
’2006LD’	1.231253797	0.100830557	24.55878256	67.76727884	187.6835581	293.0014727	54800
’2006LH’	1.084255942	0.315580823	7.850170754	95.2636774	264.6333	4.356061282	54800
’2006MA14’	2.119549271	0.492229889	7.424559935	311.6921962	330.5821286	281.7563939	54800
’2006MB’	1.11369185	0.082956024	14.42846152	257.6770271	287.0397476	101.257835	54800
’2006MH10’	1.245982416	0.15580503	13.34576666	87.95384344	253.6442866	214.0744523	54800
’2006MX13’	2.174016221	0.465970866	5.735762267	8.383306537	318.635341	255.2013582	54800
’2006NL’	0.847767992	0.575896332	20.08202853	115.2657005	29.32921267	129.8814966	54800
’2006NM’	2.784584839	0.615385299	14.20152343	292.6695372	29.98806346	175.3310008	54800
’2006OC5’	2.399910822	0.651793585	4.747083364	149.2591683	245.6744352	208.5467704	54800

’2006OD5’	2.668175099	0.545870332	10.38056615	143.8690114	223.6492328	179.3497276	54800
’2006OD7’	1.334852457	0.166253014	30.33365774	127.7107541	197.2587982	171.0510937	54800
’2006OE7’	1.547850412	0.780156336	10.414387	291.7289905	150.5767201	51.31934783	54800
’2006OF5’	2.75302244	0.537273288	10.16290175	137.4294015	194.9203083	182.2643647	54800
’2006OG15’	2.545107798	0.626722666	17.15763354	119.1792864	264.065678	190.3974685	54800
’2006OH15’	1.511486498	0.294556581	37.004071	122.9662361	76.85190146	166.0992536	54800
’2006OS5’	2.864328901	0.591097949	26.46025408	293.2239492	58.29680961	163.6026765	54800
’2006OS9’	2.746933613	0.902186238	21.14856894	127.5038679	35.90244216	195.5157254	54800
’2006OY4’	2.363811248	0.539122183	2.94679346	84.47106884	231.5486826	229.4075979	54800
’2006OZ’	2.201750318	0.466287564	5.010680316	40.53336435	318.6869872	239.9893936	54800
’2006OZ4’	1.022108497	0.432487916	17.47928204	278.5750642	141.5198452	25.42605409	54800
’2006PA1’	2.033904737	0.550238105	2.433140204	322.5536789	85.51849869	253.5613957	54800
’2006QB31’	1.176944608	0.611461579	24.72030591	341.0745644	228.4237697	327.8922054	54800
’2006QE89’	1.954172617	0.565149956	20.97504854	319.7848729	75.76779342	281.3993566	54800
’2006QJ65’	2.646226332	0.684395276	5.090423456	153.0460956	266.5436191	173.0405718	54800
’2006QK33’	2.548107346	0.587534083	14.66776516	150.6658019	192.912184	198.3644702	54800
’2006QM33’	1.800327901	0.310015208	40.27910837	154.9073108	174.0600729	340.0551434	54800
’2006QQ56’	0.985454897	0.045719555	2.79597911	161.2329568	331.3901577	312.660504	54800
’2006QS23’	1.192665382	0.502454647	19.96872079	154.885208	303.6606141	206.3786001	54800
’2006QS89’	1.778444711	0.36568792	16.60062946	159.2692288	229.9543973	318.3755399	54800
’2006QU89’	1.530353277	0.274382296	14.78588787	162.2747018	79.24227322	133.523375	54800
’2006QV89’	1.19172699	0.224314088	1.069767667	166.1204697	236.63764	219.7541017	54800
’2006QW89’	2.618395645	0.566354708	4.87333258	125.5234857	136.341494	218.3862233	54800
’2006QX5’	2.122376985	0.453968931	5.988326424	326.2325874	6.394761856	264.0211637	54800
’2006QY110’	1.925256332	0.572953678	6.407027143	222.7565135	221.4366656	280.9197321	54800
’2006RA55’	2.431495737	0.50100709	1.808537372	22.17435492	60.58185505	182.7608637	54800
’2006RG7’	1.885015127	0.350225363	22.02277574	181.0574194	191.9423221	292.4012748	54800
’2006RJ1’	0.950810788	0.300657194	1.414570232	93.49833354	110.2458011	267.3990876	54800
’2006RJ7’	1.832498734	0.580594787	1.873605041	176.8364082	251.3739061	300.6839843	54800
’2006RO36’	0.905972124	0.231075556	23.85769751	271.0041797	261.1156019	338.6253939	54800
’2006SD6’	1.515749323	0.301653003	35.74789906	169.9912062	264.8969494	11.36263027	54800
’2006SD25’	1.915715011	0.468845746	1.453769835	296.9254624	37.08558423	305.2336084	54800
’2006SK134’	1.879517511	0.536177279	34.69133014	185.3145102	74.34114234	337.0082471	54800
’2006SL198’	1.799812721	0.373750021	16.5542786	173.3041303	213.014556	306.9728854	54800
’2006SM198’	2.367704922	0.533843124	11.0669621	359.336588	345.0949172	218.4820499	54800
’2006SO19’	1.242384469	0.272029429	14.24310742	98.37915376	169.6887241	267.0836807	54800
’2006SP198’	2.933124615	0.573943497	6.928383029	192.6820155	226.4124523	143.4198224	54800
’2006SU49’	1.412738826	0.312275381	2.519757399	303.2219475	198.8596397	348.8703444	54800
’2006SV5’	1.445230795	0.293150446	4.827104388	182.2871073	208.1515306	75.78613836	54800
’2006SZ217’	1.67309491	0.285374739	29.21485431	241.4242597	162.9514037	14.13087583	54800
’2006TB’	1.562035425	0.324554904	27.56475426	169.3625709	180.8489362	45.17725137	54800
’2006TB7’	1.251341394	0.199266999	21.25521321	185.1652691	154.0342193	215.156766	54800
’2006TC1’	1.719962056	0.375658575	4.498528928	326.1532362	160.7609636	275.0150206	54800
’2006TC8’	2.152727161	0.761186035	31.5564107	204.4891945	67.29337152	263.3238534	54800
’2006TF’	2.327265901	0.561118877	3.493723879	187.8427616	182.3946062	219.2053324	54800
’2006TS7’	0.946672026	0.57988264	5.465037432	225.4444588	299.7325867	25.90696261	54800
’2006UA17’	1.371947495	0.28663932	21.90154442	207.8048078	116.6884264	149.6134212	54800
’2006UB17’	1.140682637	0.103813783	1.991072282	213.9863772	135.1303699	296.3256328	54800
’2006UE185’	2.29489218	0.490621877	1.047879212	210.6338387	183.7175542	216.7125491	54800
’2006UF’	2.455061916	0.529630213	6.946289963	207.6133179	203.6197174	191.7078638	54800
’2006UJ’	2.224314153	0.492493829	5.345156527	203.4168095	164.5829059	235.5588305	54800
’2006UJ185’	1.692325482	0.57930632	0.865046875	35.04169208	78.75841243	319.3422865	54800
’2006UK217’	1.489958762	0.668531601	41.02398703	217.3172118	311.4258937	10.73747803	54800

’2006UP217’	1.203192124	0.489614959	1.519455995	206.855916	82.92309327	258.1325178	54800
’2006UQ216’	1.103772825	0.162519994	0.472588458	217.9150647	247.3899949	235.8873436	54800
’2006UR216’	1.678147281	0.533103261	14.25633236	21.71427557	133.9675277	313.3215898	54800
’2006UZ215’	0.89009066	0.206964159	14.27324061	35.22976853	222.1361309	296.623409	54800
’2006VB’	1.729223865	0.424762905	8.692411042	36.99291243	325.0408524	343.4127161	54800
’2006VB2’	1.791795877	0.78770041	15.82309007	4.428394512	171.4476719	284.3825001	54800
’2006VB3’	2.836130403	0.547258374	11.02211015	46.47506651	336.5443679	160.7832907	54800
’2006VB14’	0.766821681	0.421307727	31.02541492	258.7717793	346.4470722	203.1290423	54800
’2006VB45’	1.222441519	0.164707863	12.48099262	234.5428409	171.1840133	189.0013131	54800
’2006VD2’	2.577938062	0.599865646	9.022388233	121.1397404	241.794985	187.2560684	54800
’2006VG13’	0.817939446	0.303345126	5.854231573	96.67287587	115.1605395	122.3568532	54800
’2006VP13’	1.177770873	0.142644229	11.07283187	231.010284	234.5349395	174.8591478	54800
’2006VT2’	1.262674247	0.723171739	31.78993998	59.51511853	152.5547336	72.53237287	54800
’2006VU2’	2.316964461	0.548663644	2.173569901	226.823137	172.4391752	211.3510049	54800
’2006VV2’	2.389881623	0.603714609	23.65725163	9.935563893	144.9670781	169.6907599	54800
’2006VW2’	1.235960556	0.294490276	10.04497069	229.8913284	299.5668283	89.3461397	54800
’2006VY2’	0.892462204	0.377137019	14.56326938	231.2981902	327.0146165	41.57254825	54800
’2006WA30’	1.628852375	0.422160977	10.06676053	241.4951203	226.860636	330.942649	54800
’2006WB’	0.84961229	0.180549435	4.908516557	65.41067387	162.4968319	49.31903802	54800
’2006WB30’	1.643436212	0.360741683	3.63398328	70.57043181	359.9510218	340.8900983	54800
’2006WD129’	1.881849475	0.500911092	6.02576927	56.80474233	319.4175121	295.0226574	54800
’2006WG130’	2.399000021	0.598304328	11.26306765	64.05241626	325.3601308	202.0039475	54800
’2006WH130’	1.304406277	0.180971955	17.04390405	66.98026285	18.55043745	110.3691598	54800
’2006WJ3’	1.752400442	0.424104156	15.35503744	232.6456738	178.3817123	315.3102376	54800
’2006WN1’	2.099443305	0.450705984	4.014308265	239.4308976	93.3387461	286.716832	54800
’2006WP127’	2.531892571	0.767345938	6.079887514	178.3330709	22.93000819	142.8404874	54800
’2006WQ29’	1.600832966	0.393570882	8.068607614	112.0762234	136.4592391	172.834271	54800
’2006WQ127’	1.309169992	0.508540883	18.10619931	73.56098187	87.41163329	84.50998484	54800
’2006WR1’	1.333117208	0.614106632	38.1624608	106.0672876	132.0439543	67.07909408	54800
’2006WR127’	0.907119585	0.375986508	16.78751565	260.8583563	351.4176134	306.2651562	54800
’2006WT1’	2.469661228	0.601241015	13.68580215	244.957998	170.5775564	186.7974544	54800
’2006WY3’	2.423446739	0.647095196	2.74088052	9.335740258	323.4936429	212.7342954	54800
’2006WZ2’	1.694544037	0.329974593	24.6586013	354.4543372	65.90764841	10.34215255	54800
’2006WZ3’	1.746578014	0.577282882	3.804998815	176.3713114	0.134495289	271.3747804	54800
’2006XH1’	2.362614477	0.454878816	6.319127687	279.0854331	183.4062494	189.1967815	54800
’2006XN4’	2.603140042	0.607849962	6.65544728	260.386809	191.850022	166.5409523	54800
’2006XP4’	0.872500426	0.213872489	0.537739623	296.7701806	343.2993675	307.2566054	54800
’2006XY’	1.498001046	0.338512791	3.638484111	257.9691656	184.0974253	24.79984094	54800
’2006YA’	1.737719194	0.424334581	15.44316903	92.23449892	28.01365623	292.9093092	54800
’2006YF13’	0.918934057	0.403488502	10.53118451	205.358165	95.33333973	189.0496185	54800
’2006YN’	1.477493664	0.224887705	15.27411126	106.3289503	307.7857475	56.01580078	54800
’2006YP44’	2.542698266	0.626222183	1.985118004	88.23329803	333.0661766	177.2104423	54800
’2007AC12’	2.775518896	0.548407195	24.05151827	51.55880208	2.675366186	145.8294473	54800
’2007AH12’	2.044242667	0.452866091	10.45366284	126.6153821	0.604966896	227.8497152	54800
’2007AM’	0.798648387	0.467224575	11.72110504	107.1797863	172.4933142	63.33833118	54800
’2007AS2’	2.585833562	0.62394746	3.76034803	308.4445598	121.5087543	170.5663206	54800
’2007AV2’	1.437535517	0.473701333	12.29728117	286.0276947	293.185354	342.2053031	54800
’2007BG49’	1.843958728	0.320968895	7.8946103	332.947098	281.4162406	171.6280808	54800
’2007BJ’	3.06727377	0.692339565	44.35250884	298.672498	134.1870175	131.569434	54800
’2007BT2’	1.632399446	0.223957098	26.85446764	31.05304096	148.8763195	306.8551926	54800
’2007BT7’	2.171285778	0.539748028	17.43045477	294.8923097	117.0216004	225.37321	54800
’2007CC27’	1.684629529	0.491495929	2.117518171	324.1549499	125.1803482	313.5213163	54800
’2007CF19’	2.985697707	0.630667215	18.1132383	321.509937	244.4344299	110.55095	54800

’2007CO5’	1.677547276	0.308621232	47.94189511	138.5095459	327.9829686	316.177715	54800
’2007CO26’	2.78409828	0.624799695	6.277248947	357.2570224	137.8473783	139.6078957	54800
’2007CP5’	1.65371182	0.357843081	35.96811264	137.9427595	41.38987582	288.1789724	54800
’2007CR5’	1.617959624	0.44472566	13.86482378	138.8789791	310.687426	334.6663997	54800
’2007DB83’	1.750625393	0.300043554	10.90336853	55.3230499	162.1622124	240.9282545	54800
’2007DC’	1.351274519	0.324137148	0.408091443	174.8606397	278.1484904	77.92822406	54800
’2007DD49’	2.172503027	0.605843992	8.196535125	150.2908322	277.1541425	216.3446906	54800
’2007DH8’	1.436981284	0.263427573	4.450618649	149.5664073	356.11921	12.97741998	54800
’2007DJ8’	1.629340837	0.361828594	29.77796159	332.1452112	154.697351	318.3958964	54800
’2007DS7’	1.180031036	0.399866599	8.448132357	148.5086811	270.7843811	185.3935489	54800
’2007DS84’	1.866713661	0.445958685	8.908533624	30.69844086	172.5747001	230.8607066	54800
’2007DX40’	1.533972157	0.536960414	0.448723587	329.7041592	273.9220554	304.612159	54800
’2007EF88’	2.497306571	0.573760961	7.108208541	167.3874614	323.7484027	164.4829314	54800
’2007EH’	2.165849076	0.657245791	1.39020461	352.7776428	246.0480873	181.6143399	54800
’2007EK’	1.126257843	0.272302628	1.206217168	168.5849095	83.24921443	107.327271	54800
’2007EL26’	1.271107811	0.115316897	13.71680818	344.3689348	229.422073	33.23144389	54800
’2007EQ’	1.629086372	0.447898206	5.669301717	106.4866372	3.10329487	322.3499389	54800
’2007ES’	1.579834767	0.601365175	35.54411322	334.4406722	19.68928734	49.86073397	54800
’2007EU’	1.445506709	0.307104785	13.25786172	164.1383001	334.5572523	13.27868242	54800
’2007FD3’	2.219685437	0.533924331	33.84385153	20.6918452	140.482573	189.9516024	54800
’2007FH1’	1.728571222	0.386428102	22.77208332	0.75868594	228.6962811	243.3123878	54800
’2007FJ1’	1.798568671	0.401971687	3.331615286	130.4200824	50.11110292	252.7425224	54800
’2007FO3’	1.270609216	0.297497788	6.293306091	356.765837	262.1040832	18.48273522	54800
’2007FS3’	1.585608948	0.41819402	3.22175153	179.6049051	313.3796013	324.9713811	54800
’2007FY20’	1.458614853	0.386868106	5.375789335	11.98234094	230.2721879	317.5925649	54800
’2007GF’	1.300663605	0.378640386	18.87737941	59.04640367	32.95127481	103.7412786	54800
’2007HB’	2.10823246	0.397901307	23.27144669	67.46570449	103.7720264	207.3920208	54800
’2007HD70’	2.115168544	0.473425121	5.658167766	161.4293641	82.0559141	180.097391	54800
’2007HG44’	2.470037661	0.720023539	8.422351812	64.79827424	33.0438018	168.2632751	54800
’2007HH44’	2.420829123	0.569983719	1.739142758	173.2712038	71.47909199	145.9548138	54800
’2007HL4’	1.118522081	0.089709231	6.530351486	31.18720817	138.5392778	166.3018148	54800
’2007HW4’	1.484073705	0.766565583	1.239332826	138.8862389	196.9122752	291.2493954	54800
’2007JB21’	0.986667896	0.109277756	13.46197314	227.9939011	250.8464774	311.4221456	54800
’2007JD’	2.83306509	0.812853997	12.26481964	228.808353	93.15656162	108.2930371	54800
’2007JF16’	2.022564321	0.675637328	44.00760037	225.5099097	221.1754762	257.0549672	54800
’2007JW9’	2.023731585	0.439925287	2.491717565	230.9542524	334.9259298	203.0443375	54800
’2007JX2’	1.707546488	0.526955543	4.221370501	44.55430045	87.66212359	280.4330552	54800
’2007KE4’	2.383196467	0.571578632	9.343543269	65.08426193	194.5849301	144.9423391	54800
’2007KF7’	1.717676214	0.385962885	11.85326963	64.02500986	185.6334046	240.4377952	54800
’2007KV2’	1.11632286	0.312769194	13.72167824	235.5601459	264.491845	169.8417237	54800
’2007LA’	1.550825886	0.593680397	33.51841382	245.5724562	107.6731495	245.7646201	54800
’2007LE’	1.83852635	0.51668429	29.48608655	73.90117497	119.9223219	231.4752101	54800
’2007LF’	1.68283586	0.419812554	6.981243974	239.5526699	333.6119419	260.3951865	54800
’2007LQ19’	2.609659781	0.62774943	17.05579483	110.9413406	207.5953158	121.027545	54800
’2007LS’	2.694684706	0.682072835	6.36939053	201.0048722	168.5955558	79.01705424	54800
’2007LT’	1.496856395	0.378920217	0.682602743	222.3363131	342.6334937	315.792791	54800
’2007LV’	1.762718999	0.270774125	16.99820929	70.37621067	261.9513437	192.1169164	54800
’2007LW19’	2.357875093	0.581992294	2.128998432	63.82209805	232.9659631	137.2842148	54800
’2007MJ13’	1.448605608	0.383201631	10.80281238	267.9833425	58.23015959	272.1953867	54800
’2007ML13’	1.297464646	0.08497482	18.01433942	135.5695882	139.2094147	343.4114004	54800
’2007ML24’	0.75828218	0.358956827	33.43286044	281.8927526	201.4812849	275.3349915	54800
’2007MT20’	1.845387724	0.612916086	16.60232098	226.5178821	148.4209891	177.0713842	54800
’2007NL1’	1.239683799	0.249129648	18.63412425	117.1066059	266.8686858	302.3565448	54800

'2007NS4'	1.874004889	0.597705586	5.80536938	11.28469956	47.18948194	146.2145341	54800
'2007OH3'	1.947705317	0.460907624	8.306797112	296.4402716	3.677290323	179.8792303	54800
'2007OR9'	1.625557887	0.262784492	11.81014183	138.4343068	125.5096303	258.6564139	54800
'2007OV'	2.478797655	0.483788124	12.67453921	353.7042749	333.1930003	120.3363419	54800
'2007PA8'	2.829249103	0.661501058	1.985441888	143.0410926	291.877141	57.33930216	54800
'2007PF6'	1.298562939	0.416285266	25.60752297	316.4493405	251.1800857	16.45270586	54800
'2007PH25'	2.536268493	0.815411473	53.24089606	150.3744608	331.8075507	92.66795391	54800
'2007PQ9'	1.426439656	0.241156233	8.556541716	136.0339351	215.3455076	256.9668914	54800
'2007PR10'	1.233086911	0.892680914	21.00051578	335.1761107	190.6476199	20.29633437	54800
'2007QA2'	2.157363356	0.435339089	4.372302492	298.7374424	22.88685329	146.6999421	54800
'2007RA9'	2.678571188	0.54086745	14.43919033	119.5786256	256.0950954	94.95609595	54800
'2007RE2'	2.289211137	0.535613126	12.28393929	160.6054378	188.488376	127.1626275	54800
'2007RO1'	2.262937585	0.605553152	8.513040703	339.6024245	306.886037	142.7531464	54800
'2007RP9'	1.53800245	0.384458816	26.38059927	342.3336404	61.83938418	205.1173679	54800
'2007RQ12'	1.792623721	0.473236926	1.239635988	84.50261666	222.2543964	196.2491465	54800
'2007RR12'	2.007787126	0.730273375	7.072835537	336.1715634	132.0828708	130.6496792	54800
'2007RT9'	1.65329669	0.566184535	20.14743079	164.0012353	24.97734786	321.2856317	54800
'2007RU19'	1.715400417	0.405542153	2.833467612	353.1757545	340.5203551	201.2981242	54800
'2007RX8'	1.138499401	0.211343191	9.398224463	348.4986059	281.9350661	58.74635598	54800
'2007RY9'	1.269430917	0.152538359	28.50448247	169.3093744	192.9360481	296.3771154	54800
'2007RZ8'	1.352420667	0.186302178	10.27161109	343.8222675	7.021833858	278.973814	54800
'2007SU1'	2.40659784	0.590275692	2.476128697	355.0368648	30.16526621	109.0222633	54800
'2007SV11'	1.734413072	0.493704836	32.23850476	292.2981879	23.63623413	193.3149946	54800
'2007TB14'	2.493277444	0.637506951	5.976945829	200.7617795	124.4018539	114.3560479	54800
'2007TC14'	2.096929392	0.808206199	4.644703779	224.2492114	269.1199149	118.2899593	54800
'2007TD71'	1.286333518	0.279593393	49.75589917	49.61260458	220.7340433	349.3213899	54800
'2007TE66'	1.05557762	0.204382795	10.24499946	195.7904661	355.1768915	220.516231	54800
'2007TG15'	2.253705023	0.475169932	4.594521509	282.3283869	136.5514374	107.0657483	54800
'2007TG71'	1.305012429	0.238403098	10.40170752	18.3452764	44.44703599	246.1310004	54800
'2007TH71'	1.341208071	0.313057601	7.064043465	20.04760743	52.85548056	233.6718453	54800
'2007TK15'	1.909119252	0.424901006	1.042620055	114.7126717	205.4171005	183.7230589	54800
'2007TL15'	1.39283806	0.547741321	1.330676137	29.76274272	240.7952401	292.3247767	54800
'2007TS68'	2.248072304	0.436821825	5.581205597	167.0427429	137.1395435	161.4249431	54800
'2007TX18'	2.137378099	0.416174728	7.371593217	284.2906693	15.7513646	182.8151343	54800
'2007TX24'	2.314328363	0.535643589	8.159611711	200.2837045	168.8023816	118.611723	54800
'2007TY18'	2.154633992	0.409461268	8.048347993	6.239170699	301.5156778	173.7045273	54800
'2007TY24'	2.506299304	0.579322629	4.293164905	194.2807837	180.7836589	103.9416347	54800
'2007UH'	1.172567994	0.336185198	15.13137213	206.6954498	294.7868992	241.7299433	54800
'2007UJ'	1.144769183	0.140323823	22.69212217	22.64883991	42.7652858	298.1103415	54800
'2007US'	0.957493067	0.575268828	12.07794427	24.32911406	202.7610585	167.913763	54800
'2007US3'	1.51270674	0.363556305	24.34064841	28.83542529	286.9322556	251.2224904	54800
'2007US6'	2.220799022	0.447838752	12.42755595	225.7557974	224.9165802	97.84082538	54800
'2007US12'	0.906035282	0.514565398	8.804662769	211.1968373	44.54922183	181.8649695	54800
'2007US51'	2.192040946	0.632382445	1.472272772	39.36231325	297.9271399	132.4052108	54800
'2007US65'	2.644177985	0.520638028	1.020722358	192.509018	123.5990021	121.0591939	54800
'2007VA3'	2.449360189	0.610198687	2.891910092	222.1534771	216.5063018	93.01477238	54800
'2007VA188'	2.973260871	0.63161949	20.24767148	38.55643537	333.0930587	77.50715384	54800
'2007VB138'	0.77250904	0.430964363	6.022108481	42.22609962	161.4468168	72.00952061	54800
'2007VD3'	0.97221917	0.150166153	13.12086093	53.85004856	179.7641877	206.8531362	54800
'2007VD8'	2.297693614	0.594142074	3.155759105	43.5795438	318.6038973	119.3665615	54800
'2007VD184'	1.942499802	0.503907555	1.232176362	239.5340031	204.2777996	127.7795517	54800
'2007VE138'	1.357518105	0.418246649	19.34053503	50.33988065	73.52214323	206.0241494	54800
'2007VF191'	1.912451478	0.410672179	11.0083166	67.21741755	318.5902666	153.3731325	54800

'2007VG'	1.956930448	0.664892871	51.15978761	33.51652054	261.5823124	166.0169226	54800
'2007VG3'	3.293698038	0.693953659	10.95047274	215.4101566	174.8559757	66.10232595	54800
'2007VH3'	1.970690089	0.433199993	2.91909582	120.319282	345.133154	116.0572552	54800
'2007VH186'	1.574055043	0.196993074	19.23984794	51.53258328	15.06271163	172.858764	54800
'2007VJ184'	1.686002002	0.461233605	18.17286176	41.5581464	62.22266313	150.5343663	54800
'2007VL3'	1.452297994	0.451823243	2.331225363	225.3658111	101.3748656	250.8540209	54800
'2007VL184'	1.294427803	0.207324348	27.65478504	48.27215298	348.0861891	264.9149588	54800
'2007VL243'	0.965286383	0.728625913	43.33757724	114.7944596	91.21986989	310.5471227	54800
'2007VM84'	1.298159065	0.189768839	24.62916732	44.6509187	7.624259961	254.6391482	54800
'2007VN243'	2.150489287	0.612513923	4.449511553	91.03317179	51.93331203	94.4052854	54800
'2007VO84'	1.308348674	0.185911498	28.30774397	47.61108026	302.4613424	295.0192978	54800
'2007VQ4'	2.635066551	0.517394437	26.53924779	59.58246647	99.27717772	62.23172915	54800
'2007VS6'	1.23571105	0.44654899	7.954316793	220.3556309	83.31000859	326.8464717	54800
'2007VV6'	1.415119416	0.279995062	3.804350502	235.8067446	149.8020425	237.5978676	54800
'2007VW137'	2.231328744	0.737208011	5.957937034	300.0794274	244.6333652	74.22943057	54800
'2007VX6'	2.267268359	0.614216671	41.11905192	233.8241786	119.8196712	126.1566287	54800
'2007VZ30'	1.604647953	0.194819797	2.454626177	344.749558	99.12134029	161.1674235	54800
'2007WB'	1.669554599	0.709624877	13.75725971	83.158484	203.6400547	220.3377708	54800
'2007WC5'	0.973226773	0.210437117	8.537132306	236.8206652	66.24389089	116.9103448	54800
'2007WD5'	2.464426669	0.597564294	2.423168216	68.40153064	309.7961721	104.3621911	54800
'2007WE55'	1.910944073	0.572810581	11.51873398	304.0480155	205.6783868	113.0341646	54800
'2007WV3'	1.667837041	0.452503524	12.65840321	261.4713026	107.6294309	200.544185	54800
'2007WX4'	1.775220197	0.301982627	28.17260695	254.4791733	95.14656622	185.5574056	54800
'2007XA10'	1.609388698	0.355474927	19.91468699	75.53736154	314.8942353	195.0003802	54800
'2007XA23'	1.60586259	0.376497597	8.327720985	78.98656551	62.49575295	142.9201536	54800
'2007XD10'	2.157271162	0.665587276	18.57820958	270.6024107	79.69382327	134.1907164	54800
'2007XF18'	1.480387915	0.481510511	59.51821295	76.16757981	267.9644745	232.6979738	54800
'2007XH16'	1.186989723	0.234809621	27.43074482	91.33350354	58.25716951	222.7624941	54800
'2007XJ16'	2.256484479	0.556535836	6.297846018	309.5651962	32.09293876	147.7755113	54800
'2007XJ20'	1.693188146	0.599508679	10.62436131	56.73360802	151.7083411	96.01372314	54800
'2007XO'	1.563567116	0.719271454	14.69400674	69.76844839	113.2015737	156.8052418	54800
'2007YF'	0.95360344	0.119659911	1.652385495	277.5514384	34.5252577	128.0214789	54800
'2007YH'	2.153491941	0.618080029	29.4573937	273.576748	96.90727876	127.2861927	54800
'2007YM'	2.578168909	0.61632854	0.984582859	59.42424484	11.51159182	85.72035646	54800
'2007YN1'	2.690771685	0.718449697	3.90653181	84.7732121	294.4597396	87.05107356	54800
'2007YO56'	1.280521229	0.357202381	15.61016928	287.7356273	333.3036087	101.072922	54800
'2007YQ56'	1.140641469	0.287920736	26.4573077	276.0746386	273.0192912	201.5403444	54800
'2007YR56'	2.01237996	0.515769179	10.32824699	97.60386523	336.1246553	122.443851	54800
'2007YT56'	1.294592673	0.287662127	5.998999151	302.5658385	81.66273946	270.8455768	54800
'2008AD'	1.706023464	0.679073101	6.920108178	224.075113	338.0503533	119.6078055	54800
'2008AE4'	2.326470909	0.561831614	5.539920344	101.8767383	28.33473922	84.86721693	54800
'2008AF4'	1.382540706	0.410694168	8.919727694	109.4292902	293.3278365	227.7729414	54800
'2008AG1'	1.787412857	0.560500445	4.241805421	196.4035132	204.5063912	156.4966293	54800
'2008AJ33'	3.28512867	0.655499596	11.09871134	98.42090335	308.7490331	61.71081487	54800
'2008AM33'	1.862589801	0.411522381	1.515023104	307.9939495	222.2747188	105.5300833	54800
'2008AP33'	1.338206272	0.481273034	8.246621839	122.5607835	257.3886517	243.7530714	54800
'2008AS28'	2.428162893	0.731733951	19.90079699	86.51117382	248.1213475	106.7581036	54800
'2008BC'	2.419258416	0.600116669	14.28144676	291.9567846	157.6929027	88.93921112	54800
'2008BC15'	2.167138632	0.709670421	3.386974465	309.8093837	263.7515753	79.98286763	54800
'2008BO16'	2.434054225	0.808143172	8.609630115	134.0049856	254.2734126	92.24990551	54800
'2008CA5'	1.861567712	0.593167684	24.99156424	331.6794492	272.8771896	87.76233008	54800
'2008CA6'	1.709632507	0.468691211	5.917158314	309.9967229	271.2842031	90.53151249	54800
'2008CC6'	1.262448857	0.774127319	9.141745128	146.9312858	136.4296397	161.4387447	54800

’2008CC175’	0.954383283	0.499978489	10.12622279	125.6337653	166.4679461	218.7321185	54800
’2008CD22’	1.576142098	0.322310159	10.17650935	148.5552279	8.952620854	137.4356002	54800
’2008CE119’	1.212180465	0.178256933	7.771383277	147.504193	49.59763509	177.0805723	54800
’2008CJ22’	1.44310092	0.278650814	20.47256136	140.0391723	332.1646497	183.0007862	54800
’2008CL20’	0.766442962	0.319530169	15.94225813	321.3409761	349.6496704	269.8885309	54800
’2008CL70’	2.180239066	0.671548973	21.36392078	144.2473943	274.7044004	106.8216128	54800
’2008CM116’	1.630429942	0.663318458	18.71329071	0.172408113	355.6948811	260.7525768	54800
’2008CN116’	1.219558745	0.361131611	2.387944059	135.714906	87.72817959	170.551843	54800
’2008CP’	1.12055759	0.077795839	13.5964099	140.0706529	17.08539146	229.8823413	54800
’2008CP23’	2.207142823	0.455073813	3.774747701	136.9992522	82.602563	59.70315662	54800
’2008CY118’	1.303997354	0.373375062	21.63378849	321.2767357	94.13868119	239.5876458	54800
’2008DJ’	1.982405947	0.60370499	5.051221717	319.3066456	117.7893575	117.5443608	54800
’2008DL4’	0.929354524	0.122755791	3.205813721	341.3399449	42.20456848	68.62845392	54800
’2008DV22’	2.713075223	0.636633979	11.07695605	153.1227034	56.19611785	50.98729971	54800
’2008DW’	1.988995589	0.360641541	6.735948702	161.4374133	10.0817202	92.68241408	54800
’2008DW22’	1.750529948	0.382105884	0.835556902	105.7537595	75.09724143	108.1746296	54800
’2008EA32’	0.61591521	0.304956607	28.26937499	100.9931243	181.8233198	51.06663468	54800
’2008EC69’	2.752112397	0.61926715	24.80631804	93.3351654	178.1914203	38.05678141	54800
’2008EC85’	3.157340942	0.736272931	55.70314347	169.0401581	272.038644	59.12102376	54800
’2008EE9’	1.324995828	0.527769536	9.970949501	180.6006505	249.5882654	210.9307317	54800
’2008EF9’	2.146480049	0.600217534	6.636222965	152.9283711	76.60798762	68.37316787	54800
’2008EF32’	1.623214167	0.520218908	1.726550322	349.1454865	112.3355849	148.1246196	54800
’2008EL85’	1.920373104	0.56248811	2.296535479	161.5404061	310.6354014	113.0547892	54800
’2008EM6’	1.844304146	0.491563114	9.155610841	165.1731696	54.80821838	88.02183854	54800
’2008EM9’	1.959678554	0.8514048	9.392273555	229.736674	181.690846	117.4217826	54800
’2008EN82’	2.50361579	0.552412076	11.98040597	207.0304846	194.6129899	132.1972664	54800
’2008EO6’	1.903418603	0.400862972	19.06933905	173.1270957	306.8250174	125.4282127	54800
’2008EP6’	1.20950206	0.292916734	17.73124183	303.3394987	130.269862	253.1352727	54800
’2008EQ’	1.7537226	0.458236668	2.723113437	168.7177571	35.52910529	101.4933384	54800
’2008EV5’	0.960444724	0.083806458	7.426780642	93.60101526	236.7545841	89.46290793	54800
’2008EV68’	1.460038719	0.284299898	3.262782337	191.3591881	288.7723483	175.0733055	54800
’2008EX5’	1.36027403	0.391321101	3.38558436	16.38604904	66.01074142	209.4401083	54800
’2008EY68’	0.745041283	0.759848542	19.7970037	175.5222829	198.9503221	143.3370246	54800
’2008FK7’	1.887503241	0.393578323	1.432459064	346.9922877	251.4781623	74.95719336	54800
’2008FY6’	1.813085867	0.387221797	25.53466904	10.22033535	106.5665519	131.589136	54800
’2008GF1’	1.22945337	0.465365416	1.421379108	16.50070711	276.1965722	128.8166435	54800
’2008GM2’	1.051831883	0.15721376	4.095999548	195.1238795	278.2255093	282.9633158	54800
’2008GP20’	1.959648903	0.441892289	32.64737505	31.62551178	92.92788036	113.514723	54800
’2008GQ3’	2.178555228	0.522881089	25.4505839	356.2047107	142.1170202	82.96430873	54800
’2008GV’	2.730474849	0.609277329	30.10145421	15.62644055	177.6074306	52.57007602	54800
’2008GX’	2.275052098	0.484815297	2.621233009	144.5625653	108.9226332	43.40416263	54800
’2008GX3’	1.888970324	0.476855519	9.497803244	196.8108125	25.06955601	81.744013	54800
’2008GX21’	1.907921515	0.415462365	9.526637466	27.05572949	166.6386476	89.00882141	54800
’2008HD3’	1.131356074	0.335715873	52.00709133	222.1019834	223.7503708	284.7091168	54800
’2008HJ’	1.631788311	0.406710292	0.927357253	47.49597957	204.0830783	88.76380074	54800
’2008HW1’	2.583037399	0.960588751	10.61353221	129.2559207	248.8705341	39.37497078	54800
’2008JF’	1.907092318	0.392831257	19.81819803	90.78388527	235.6017557	21.39246713	54800
’2008JO’	1.50889457	0.5455277	5.372054435	276.9673696	194.6588317	144.3729777	54800
’2008JO14’	2.002673272	0.796850581	4.975463805	117.1137419	349.5337476	91.78201042	54800
’2008JO20’	3.284134815	0.610176261	6.787810989	130.2816018	36.28757146	32.28801106	54800
’2008JP’	1.546614062	0.648483435	18.35809439	31.25270719	307.124833	60.44336351	54800
’2008JP24’	1.248837104	0.277408114	1.153067124	41.45630896	125.2264597	181.2563597	54800
’2008JQ14’	2.462552658	0.518163392	14.51085936	204.0133036	22.33698574	47.38435256	54800

’2008JR14’	1.645779769	0.375179275	7.857145901	231.3294143	333.1884253	105.6773888	54800
’2008JZ7’	2.656778936	0.649241227	24.23934583	219.0311563	69.65034637	29.61557778	54800
’2008KE6’	1.685134895	0.519650812	3.424037983	113.7737455	224.1729099	56.59023024	54800
’2008KS’	0.97453673	0.156802034	25.49168375	246.098733	152.7662558	47.86538376	54800
’2008LB’	2.456142477	0.608779178	4.319576087	80.087429	211.7398439	37.87195859	54800
’2008LD’	0.891852189	0.154631344	6.54205742	250.9212704	201.4273666	4.178129858	54800
’2008LG2’	0.854372601	0.228939213	3.055430677	86.18259176	336.9937773	60.29840866	54800
’2008LQ16’	1.707670581	0.736665562	7.258574338	142.1131278	17.74926191	97.05564443	54800
’2008LV16’	2.086983331	0.62361238	4.685078136	237.3326752	120.9881851	33.67219627	54800
’2008MH1’	2.694028329	0.582285308	7.981237481	304.3104772	13.62064057	20.48190844	54800
’2008MU1’	1.896236359	0.435938082	40.77987472	296.372673	56.4992014	29.77442756	54800
’2008NA’	0.961767346	0.302309265	9.920332314	104.8663316	52.84265354	245.9953794	54800
’2008NQ3’	2.451424464	0.581835174	27.17575751	109.3887541	155.2974977	43.40269809	54800
’2008NX’	1.319265811	0.206716225	0.598129452	258.2813091	25.98500327	96.39884206	54800
’2008ON8’	1.646963816	0.354010358	5.156411331	130.822436	183.7072381	54.76278606	54800
’2008ON10’	1.161686422	0.160525177	7.106839519	131.3041564	242.8682759	46.34103966	54800
’2008OO’	2.105736668	0.700305384	5.491815214	306.3560281	253.3173048	65.19785863	54800
’2008OO1’	2.425812417	0.617454752	9.33209886	304.3562482	314.1222452	42.44606769	54800
’2008OO8’	2.050597463	0.506976261	6.356049898	302.0505472	344.6692482	47.47805704	54800
’2008OV2’	2.380735925	0.574432552	15.29230942	286.4426198	307.6228254	55.41010854	54800
’2008PE1’	2.168229197	0.507142918	4.453657115	300.8900725	7.189941316	38.08315642	54800
’2008PJ9’	2.545671924	0.662160732	4.769646929	115.3562644	135.085798	40.61623577	54800
’2008PK3’	1.891756743	0.598436582	17.08578248	122.8348343	108.9833931	66.88011984	54800
’2008PW4’	1.161273593	0.272797003	2.701899557	117.7246629	294.7270868	26.06353319	54800
’2008QA1’	2.223503875	0.514754392	26.4676107	336.7427451	28.22162425	19.68063757	54800
’2008QB’	1.207174537	0.739614756	35.74936023	146.9994418	314.1526811	32.64790635	54800
’2008QC’	2.273426345	0.519523976	11.11165667	17.07348434	301.4026477	32.2592682	54800
’2008QF’	2.077135208	0.377915326	3.777087167	192.7259295	136.1343972	31.8889799	54800
’2008QU3’	0.869059743	0.247920503	14.07940354	303.6591683	241.2041754	247.2544342	54800
’2008QZ’	2.169007174	0.423625146	6.830914334	160.7240604	194.0229565	24.64717178	54800
’2008RG98’	2.188982062	0.768671439	10.7355962	340.1386126	168.3883273	336.8237457	54800
’2008RH1’	1.063860489	0.161935156	7.465780726	350.9817274	147.4113108	296.7169341	54800
’2008RJ1’	2.166900858	0.471608655	16.33060013	191.9946059	54.16898834	86.63103317	54800
’2008RS24’	2.090793267	0.613734131	17.14791858	160.1927202	264.2754397	5.625641495	54800
’2008RT24’	1.814950243	0.418141127	6.917340468	161.9680331	212.7607361	21.92179813	54800
’2008RU’	2.114250718	0.648648061	7.211176964	140.1773502	274.5346097	12.44048506	54800
’2008RV’	2.270774581	0.562366268	2.277263661	119.9514955	192.3503175	31.91358742	54800
’2008RW24’	1.750610905	0.500077787	2.160744609	14.92907675	38.70927706	12.05661767	54800
’2008RX24’	2.287079736	0.440343863	2.029875575	7.417832466	340.1425524	24.08932191	54800
’2008SH82’	2.441143392	0.590185458	4.562684748	269.1489748	157.2937743	358.6523047	54800
’2008SJ82’	2.377310798	0.578398633	9.712594169	135.3564577	218.2021546	20.21713622	54800
’2008SO’	1.33030599	0.233337606	7.136259744	191.1575901	71.88958917	119.3798452	54800
’2008SP’	1.692011642	0.350857606	16.53190553	187.3191391	129.783278	49.10066106	54800
’2008SQ1’	2.95895752	0.582359089	6.700271905	269.9742289	150.9795379	0.152669425	54800
’2008SR1’	2.374612613	0.648869392	16.72626182	179.2520117	115.4428428	30.94503776	54800
’2008SR7’	1.322124543	0.192405511	15.31285464	324.6174963	296.7278431	122.5123725	54800
’2008SS’	0.92839502	0.479226679	21.12461063	5.100756142	134.9680183	338.9212676	54800
’2008SV11’	2.614928492	0.721962445	8.294154202	15.68991031	102.8125262	341.6322364	54800
’2008SW7’	1.612654178	0.350768434	17.46239112	178.6951451	186.8139377	29.88057519	54800
’2008SW11’	1.134293879	0.408238088	7.432821845	28.92197811	206.877524	148.0939101	54800
’2008SW150’	1.786097177	0.561793676	20.0070399	39.61961362	51.86544735	351.7077562	54800
’2008TB’	2.473961851	0.60398086	27.38442773	188.0806232	209.1312873	9.075278678	54800
’2008TC’	2.063364496	0.50062374	2.677606613	211.1985136	106.4498206	34.39936545	54800

’2008TC3’	1.292904624	0.301100353	2.452792629	194.1072741	234.1540877	330.1153622	54745.91815
’2008TD4’	1.821391949	0.617560966	14.47606014	222.0772846	54.7082736	49.02723609	54800
’2008TE157’	2.73539792	0.623846507	1.975431966	26.25019507	77.3944987	350.2391927	54800
’2008TF4’	1.894053547	0.319727661	24.05473463	213.4909695	113.0926841	46.56557665	54800
’2008TP26’	1.094520404	0.287369201	13.31787308	12.38628373	241.6314361	139.3631396	54800
’2008TS10’	1.259219358	0.203115771	1.476221964	5.608952579	345.2497815	52.86292431	54800
’2008TS26’	1.427974084	0.465503938	6.32061411	16.12783428	284.8794092	59.37869443	54800
’2008TT26’	1.345510382	0.257273129	8.476473885	210.5859096	190.0007976	17.99280154	54800
’2008TX9’	1.91160662	0.510239456	7.652182712	199.9007695	223.4563775	5.251846522	54800
’2008TY3’	1.840253032	0.392200743	3.247424368	25.57575086	327.1851147	30.16361292	54800
’2008TY9’	1.432311285	0.279203856	8.28511503	205.2128496	158.5713566	36.45647621	54800
’2008UA92’	2.618284782	0.607588573	3.062866686	39.54580608	351.5995273	8.431836851	54800
’2008UA202’	1.032725759	0.068786393	0.26459756	21.10151202	300.7308988	96.41230286	54800
’2008UB7’	1.235224537	0.593548535	2.018702228	219.6975674	287.5317161	340.9273733	54800
’2008UM3’	1.459696745	0.251030049	10.7013342	213.7291519	211.7723761	0.623347951	54800
’2008UO90’	2.192815069	0.431602838	5.728998275	221.0928882	100.724262	43.66453616	54800
’2008UT2’	1.80391249	0.48313163	7.573209216	208.5770819	130.8389768	32.25619254	54800
’2008UU99’	2.32192398	0.534440552	4.25971379	44.03660517	349.8771499	9.419107424	54800
’2008UX91’	1.44415947	0.216002573	27.21159548	32.85211045	328.0878012	39.56355804	54800
’2008VC’	1.12175875	0.17292069	5.724475185	218.4820738	240.5376474	339.4378351	54800
’2008VF’	0.906073861	0.325771675	26.18710234	234.4560917	3.245130859	204.1933735	54800
’2008VG14’	2.867098938	0.564879809	10.18128278	260.1497629	112.9417379	12.83128269	54800
’2008VJ’	1.709901806	0.467051336	25.90575591	227.5539008	83.29498994	51.25551953	54800
’2008VU4’	2.371886305	0.770037672	11.97415142	291.1918845	23.16850955	23.03733836	54800
’2008WB’	1.371659075	0.088384399	43.89361778	236.939816	163.4000902	17.85742786	54800
’2008WK’	1.420893563	0.283406763	6.366828594	61.86724491	28.8359168	348.7157007	54800
’2008WL’	2.740673154	0.644533653	6.725984812	277.2749624	109.8771876	7.889515969	54800
’2008WM’	1.073813849	0.141094966	12.31381436	57.73047757	84.91027036	299.8199219	54800
’6344P-L’	2.804108172	0.66708346	4.726812201	183.611772	234.0696552	79.37913845	54800

Bibliography

[1] GTOC Portal | The Global Trajectory Optimisation Competition Portal, March 2014. URL: `http://sophia.estec.esa.int/gtoc_portal/`.

[2] Texas Advanced Computing Center - Stampede User Guide, May 2014. URL: `https://www.tacc.utexas.edu/user-services/user-guides/stampede-user-guide`.

[3] Kristina Alemany. *Design Space Pruning Heuristics and Global Optimization Method for Conceptual Design of Low-thrust Asteroid Tour Missions.* PhD thesis, Georgia Institute of Technology, 2009.

[4] Kristina Alemany and Robert D. Braun. Survey of Global Optimization Methods for Low-thrust, Multiple Asteroid Tour Missions. In *Proceedings of AAS/AIAA Space Flight Mechanics Meeting*, January 2007.

[5] David L. Applegate, Robert E. Bixby, Vasek Chvtal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2nd edition, February 2007.

[6] Brent W. Barbee, Salvatore Alfano, Elfego Pinon, Kenn Gold, and David Gaylor. Design of Spacecraft Missions to Remove Multiple Orbital Debris Objects. In *Proceedings of the Aerospace Conference, 2011 IEEE*, page 114, 2011.

[7] Brent W. Barbee, George W. Davis, and Sun-Hur Diaz. Spacecraft Trajectory Design for Tours of Multiple Small Bodies. *Advances in the Astronautical Sciences*, 135(3):2169–2188, 2009.

[8] Roger R. Bate, Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics*. Dover Publications, New York, 1971.

[9] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. American Institute of Aeronautics and Astronautics, Reston, Va., 1999.

[10] Richard Bellman. *Dynamic Programming*. Dover Publications, Mineola, N.Y, reprint edition, March 2003.

[11] Regis Bertrand, Richard Epenoy, and Benoit Meyssignac. Final Results of the 4th Global Trajectory Optimisation Competition, 2009.

[12] Regis Bertrand, Richard Epenoy, and Benoit Meyssignac. Problem Description for the 4th Global Trajectory Optimisation Competition, 2009.

[13] John T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[14] Vitali Braun, A. Lupken, S. Flegel, J. Gelhaus, M. Mockel, C. Kebschull, C. Wiedemann, and P. Vorsmann. Active Debris Removal of Multiple Priority Targets. *Advances in Space Research*, 51(9):1638–1648, 2013.

[15] M. Cerf. Multiple Space Debris Collecting Mission–Debris Selection and Trajectory Optimization. *Journal of Optimization Theory and Applications*, 156(3):761–796, March 2013.

[16] Bruce A. Conway. *Spacecraft Trajectory Optimization*. Cambridge University Press, Cambridge; New York, 2010.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, Masachusetts; London, third edition, 2009.

[18] Timothy Crain, Robert H. Bishop, Wallace Fowler, and Kenneth Rock. Interplanetary Flyby Mission Optimization Using a Hybrid Global-Local Search Method. *Journal of Spacecraft and Rockets*, 37(4):468–474, 2000.

[19] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a Large-scale Traveling-salesman Problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[20] Rina Dechter and Judea Pearl. Generalized Best-first Search Strategies and the Optimality of A*. *J. ACM*, 32(3):505–536, July 1985.

[21] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[22] Claudia DAmbrosio and Andrea Lodi. Mixed Integer Nonlinear Programming Tools: An Updated Practical Overview. *Annals of Operations Research*, 204(1):301–320, April 2013.

[23] Michel Gendreau. *An Introduction to Tabu Search.* Springer, 2003.

[24] Fred Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

[25] Fred Glover. Tabu Search–part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[26] Fred Glover. Tabu Search–part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[27] Fred Glover and Manuel Laguna. *Tabu Search.* Kluwer, Boston, Mass., 1998.

[28] R. H. Gooding. A Procedure for the Solution of Lambert's Orbital Boundary-value Problem. *Celestial Mechanics and Dynamical Astronomy*, 48(2):145–165, June 1990.

[29] I. S. Grigoriev and M. P. Zapletin. Choosing Promising Sequences of Asteroids. *Automation and Remote Control*, 74(8):1284–1296, August 2013.

[30] G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations.* Springer, New York, 2002 edition, May 2007.

[31] Pierre Hansen. The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming. In *Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy*, pages 70–145, 1986.

[32] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[33] David G. Hull. *Optimal Control Theory for Applications.* Springer, New York, 2003.

[34] Dario Izzo. 1st Act Global Trajectory Optimisation Competition: Problem Description and Summary of the Results. *Acta Astronautica*, 61(9):731–734, November 2007.

[35] Dario Izzo, Victor M. Becerra, D. R. Myatt, Slawomir J. Nasuto, and J. Mark Bishop. Search Space Pruning and Global Optimisation of Multiple Gravity Assist Spacecraft Trajectories. *Journal of Global Optimization*, 38(2):283–296, 2007.

[36] Dario Izzo, Tams Vink, Claudio Bombardelli, Stefan Brendelberger, and Simone Centuori. Automated Asteroid Selection for a Grand Tour Mission. In *58th International Astronautical Congress, Hyderabad, India*, 2007.

[37] Donald J. Kessler and Burton G. Cour-Palais. Collision Frequency of Artificial Satellites: the Creation of a Debris Belt. *Journal of Geophysical Research: Space Physics*, 83(A6):2637–2646, June 1978.

[38] Donald J. Kessler, Nicholas L. Johnson, J. C. Liou, and Mark Matney. The Kessler Syndrome: Implications to Future Space Operations. *Advances in the Astronautical Sciences*, 137(8):2010, 2010.

[39] Richard E. Korf. Linear-Space Best-First Search. *Artificial Intelligence*, 62(1):41–78, July 1993.

[40] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, Chichester West Sussex, New York, 1st edition, September 1985.

[41] J. C. Liou. An Active Debris Removal Parametric Study for Leo Environment Remediation. *Advances in Space Research*, 47(11):1865–1876, June 2011.

[42] J. C. Liou, N. L. Johnson, and N. M. Hill. Controlling the Growth of Future LEO Debris Populations With Active Debris Removal. *Acta Astronautica*, 66(56):648–653, March 2010.

[43] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics.* Springer, Berlin; New York, 2004.

[44] N. Mladenovi and P. Hansen. Variable Neighborhood Search. *Computers & Operations Research*, 24(11):1097–1100, November 1997.

[45] Alessandro Morbidelli, W. F. Bottke, Ch. Froeschle, and P. Michel. Origin and Evolution of Near-Earth Objects. *Asteroids III*, 409, 2002.

[46] M. Morimoto, H. Yamakawa, M. Yoshikawa, M. Abe, and H. Yano. Trajec-

tory Design of Multiple Asteroid Sample Return Missions. *Advances in Space Research*, 34(11):2281–2285, 2004.

[47] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, New York, NY, 1999.

[48] Jorge Nocedal and Stephen Wright. *Numerical Optimization.* Springer, New York, 2nd edition, July 2006.

[49] Cesar Ocampo. Finite Burn Maneuver Modeling for a Generalized Spacecraft Trajectory Design and Optimization System. *Annals of the New York Academy of Sciences*, 1017(1):210–233, 2004.

[50] Cesar Ocampo. Exact Impulsive to Time Optimal Finite Burn Trajectory Automation (July 2011 Draft). 2011.

[51] Cesar Ocampo, Juan S. Senent, and Jacob Williams. Theoretical Foundation of Copernicus: A Unified System for Trajectory Design and Optimization. 2010.

[52] Joris T. Olympio. Optimal Control Problem for Low-Thrust Multiple Asteroid Tour Missions. *Journal of Guidance, Control, and Dynamics*, 34(6):1709–1720, 2011.

[53] Christos H. Papadimitriou. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, Mineola, N.Y, unabridged edition, January 1998.

[54] Panos M. Pardalos and H. Edwin Romeijn. *Handbook of Global Optimization Volume 2.* Springer US, Boston, MA, 2002.

[55] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Pub. Co., Reading, Mass., 1984.

[56] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation.* Morgan Kaufmann, Burlington, MA, 2nd edition, July 2010.

[57] Howard M. Robbins. An Analytical Study of the Impulsive Approximation. *AIAA Journal*, 4(8):1417–1423, 1966.

[58] Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis, II. An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM Journal on Computing*, 6(3):563–581, September 1977.

[59] A. E. Roy. *Orbital Motion.* Institute of Physics Pub., Bristol, England; Philadelphia, 2005.

[60] Ryan P. Russell. Primer Vector Theory Applied to Global Low-Thrust Trade Studies. *Journal of Guidance, Control, and Dynamics*, 30(2):460–472, 2007.

[61] Matteo Rosa Sentinella and Lorenzo Casalino. Hybrid Evolutionary Algorithm for the Optimization of Interplanetary Trajectories. *Journal of Spacecraft and Rockets*, 46(2):365–372, 2009.

[62] Jinjun Shan and Yuan Ren. Low-Thrust Trajectory Design With Constrained Particle Swarm Optimization. *Aerospace Science and Technology*, 36:114–124, July 2014.

[63] El-Ghazali Talbi. *Metaheuristics: from Design to Implementation.* John Wiley & Sons, Hoboken, N.J., 2009.

[64] David A. Vallado and Wayne D. McClain. *Fundamentals of Astrodynamics and Applications.* Microcosm Press, Hawthorne, CA, 2013.

[65] M. Vasile and P. De Pascale. Preliminary Design of Multiple Gravity-Assist Trajectories. *Journal of Spacecraft and Rockets*, 43(4):794–805, 2006.

[66] Massimiliano L. Vasile. A Behavioral-Based Meta-Heuristic for Robust Global Trajectory Optimization. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 2056–2063. IEEE, 2007.

[67] Matthew A. Vavrina and Kathleen C. Howell. Global Low-Thrust Trajectory Optimization Through Hybridization of a Genetic Algorithm and a Direct Method. In *AIAA/AAS Astrodynamics Specialist Conference, AIAA*, volume 6614, 2008.

[68] Nguyen X. Vinh, Elmer G. Gilbert, Robert M. Howe, Donglong Sheu, and Ping Lu. Reachable Domain for Interception at Hyperbolic Speeds. *Acta Astronautica*, 35(1):1–8, January 1995.

[69] Tams Vink, Dario Izzo, and Claudio Bombardelli. Benchmarking Different Global Optimisation Techniques for Preliminary Space Trajectory Design. In *58th International Astronautical Congress, International Astronautical Federation (IAF)*, 2007.

[70] Changxuan Wen, Yushan Zhao, Peng Shi, and Zhang Hao. Orbital Accessibility Problem for Spacecraft with a Single Impulse. *Journal of Guidance, Control, and Dynamics*, 0(0):1–12, January 2014.

[71] Jacob Williams, Juan S. Senent, Cesar Ocampo, Ravi Mathur, and Elizabeth C. Davis. Overview and Software Architecture of the Copernicus Trajectory Design and Optimization System. 2010.

[72] Byoungsam Woo, Victoria L. Coverstone, and Michael Cupples. Low-Thrust Trajectory Optimization Procedure for Gravity-Assist, Outer-Planet Missions. *Journal of Spacecraft and Rockets*, 43(1):121–129, 2006.

[73] Dan Xue, Junfeng Li, Hexi Baoyin, and Fanghua Jiang. Reachable Domain for

Spacecraft With a Single Impulse. *Journal of Guidance, Control, and Dynamics*, 33(3):934–942, 2010.

Vita

Gregory Phillip Johnson was born to Don and Sharon Johnson on January 17, 1983 in Dallas, Texas. He completed his Bachelor of Science in Aerospace Engineering (with high honors) at The University of Texas at Austin in 2005, continuing on to the graduate aerospace program at UT to obtain his Master of Science in Engineering in 2007, and now his Ph.D. in 2014. During his time as a graduate student, Gregory also worked full time at the Texas Advanced Computing Center.

Permanent Address: gregjohnson@utexas.edu

This dissertation was typeset with LaTeX 2_ε[1] by the author.

---

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, and Ayman El-Khashab.