Ground Stations Scheduling with Genetic Algorithm

Junzi Sun

SUPERVISED BY

Fatos Xhafa, and Ed Chester

Universitat Politècnica de Catalunya
Master in Aerospace Science & Technology
July 2009
Ground Stations Scheduling with Genetic Algorithm

BY
Junzi Sun

DIPLOMA THESIS FOR DEGREE
Master in Aerospace Science and Technology

AT
Universitat Politècnica de Catalunya

SUPERVISED BY:
Fatos Xhafa
Department of Languages and Informatic Systems

Ed Chester
Aerospace Research and Technology Centre
This work presents a Genetic Algorithm (GA) approach to Ground Station (GS) and Spacecraft (SC) Scheduling problem, which is based on the space missions and ground stations from ESA (European Space Agency).

Genetic Algorithm has been used for optimization for many years. The first part of the work is to study how GA has been developed and put in position to science and engineering field. A general GA process is been introduced in this section, which describes basic operations of encoding, mutation, crossover, and selection. There are strengths and limitations of Genetic Algorithms for optimization, which are describe in this section too.

The GS-SC scheduling problem is a highly resource-constrained. So in section 1.2, the concept of Resource-Constrained Schedule is studied and defined. And the difficulties of this kind of schedule are presented.

The second part of the work defines the basic concepts of ground stations and spacecrafts, which is based on ESA examples. A mathematical model of Ground Stations and spacecrafts is built based on the definitions and assumption of the system. It is simplified so that it can be understood and modeled easily. There are three parts of the model: inputs, outputs and intermediate parameters. The system is to take the input data of spacecraft access windows and time requirements, and using an algorithm to generate a valid schedule solution.

STK (Satellite Tool Kit) is been selected for data generation of this work. Space mission of selected ones are simulated and executed. The STK generates one of the important input data: Access Window information of GSs to all SCs. Together with defined mission requirement data, they are converted and stored in the schedule system using a pre-defined structure, which is waiting for further GA process.

The GA process is the core chapter in this work. It describes the most important part of work that is approaching the solution of the entire problem. It starts from the encoding method, where two encoding methods are invested and tested, binary vector encoding and decimal vector encoding. It has been proved in this work that the decimal encoding has a better performance and computation speed than the other one. There are advantages and weaknesses that are both examined. Also crossover and mutation methods are introduced.

The focus of this designated GA is on designing its fitness functions. This task is related with the constraints and objectives for the ground stations and space mission requirements. A technique of Fitness Modules (FM) is been developed to satisfying the varieties of mission objects. Those modules can be sequential or parallel in the fitness evaluation process. The introducing of FM concept gives the answer to add and remove mission objectives without affecting the existing GA fitness functions. Thus the final evaluating fitness is by summarizing all FMs with different weights. In this simplified model four FMs that represent four mission objectives are designed, these are, Fitness for Spacecraft Access Windows, Fitness for Communication
Clashes, Fitness for Communication Time Requirement, and Fitness for Maximizing Ground Station Usage.

Every GA needs a selection method of choosing chromosomes for population reproduction. There are some traditional selection methods, which are selected, described and studied. Also we have proposed a combinational selection method to accelerate the population fitting value.

The last part of the work is to simulate the entire process in computer environment. Matlab is selected because of its excellent mathematical calculations capability. The GA is coded and executed with multiple times, in order to get the average results. Those data are all been illustrated. And one of the best schedules is been generated as the solution of the problem. The designed GA solved defined problem successfully.

In the end, the weakness of this GA is mentioned, and future work direction is pointed out.
# Table of Contents

**INTRODUCTION** ............................................................................................................. 13

**CHAPTER 1 BACKGROUND OF GA AND SCHEDULE PROBLEM** ......................... 15

1.1 Genetic Algorithms ........................................................................................................ 15
  1.1.1 A Brief History of GA ................................................................................................. 15
  1.1.2 What is a Genetic Algorithm ....................................................................................... 15
  1.1.3 Strengths .................................................................................................................... 17
  1.1.4 Limitations .................................................................................................................. 18

1.2 Resource-Constrained Schedule Problems .................................................................... 19
  1.2.1 Definition .................................................................................................................... 19
  1.2.2 Why it is hard to schedule ........................................................................................ 20

**CHAPTER 2 GROUND STATIONS, SPACECRAFTS AND MATHEMATICAL MODELING** .................................................................................................................. 21

2.1 Ground Stations .............................................................................................................. 21
  2.1.1 What is a Ground station ............................................................................................ 21
  2.1.2 The Visibility of a ground station ............................................................................. 21
  2.1.3 The Distribution of Ground Stations ......................................................................... 22

2.2 Spacecraft ...................................................................................................................... 23
  2.2.1 Spacecraft and their missions ................................................................................. 23
  2.2.2 Spacecraft Ground Tracks ....................................................................................... 26

2.3 The Complexities of Ground Station Scheduling ............................................................ 27
  2.3.1 Limited Ground Stations ......................................................................................... 27
  2.3.2 Complicated Mission Requirements ..................................................................... 27
  2.3.3 Spacecraft Visibility Clash ..................................................................................... 28

2.4 Mathematical model ........................................................................................................ 28
  2.4.1 Definitions and Assumptions .................................................................................... 29
  2.4.2 Mathematical Modeling .......................................................................................... 29

**CHAPTER 3 SIMULATION AND DATA GENERATION** ...................................................... 31

3.1 Mission simulation using STK .......................................................................................... 31

3.2 Data generation using STK ............................................................................................. 32
  3.2.1 Single ground station scenario ................................................................................. 32
  3.2.2 Multiple GS scenario ............................................................................................... 33

3.3 Data Pre-Process ............................................................................................................. 34
  3.3.1 Visibility Windows ..................................................................................................... 34
  3.3.2 Time Requirements ................................................................................................. 34
  3.3.3 Benchmark ............................................................................................................... 35

**CHAPTER 4 GA PROCESS** .............................................................................................. 39

4.1 Encoding ......................................................................................................................... 39
  4.1.1 Decimal vector encoding ......................................................................................... 39
  4.1.2 Binary vector encoding ............................................................................................ 39

4.2 Initial population ............................................................................................................. 40

4.3 Crossover for Reproduction ........................................................................................... 41

4.4 Mutation ......................................................................................................................... 41
4.5 Feasibility of the offspring solutions ................................................................. 42
4.6 Fitness, multi-objectives ..................................................................................... 43
  4.6.1 Objective 1 - Access Windows ................................................................. 44
  4.6.2 Objective 2 - Communication Clashes ...................................................... 45
  4.6.3 Objective 3 - Time Requirement of Communications ............................. 45
  4.6.4 Objective 4 - Maximize GS Usage .......................................................... 47
4.7 Selection Method .............................................................................................. 47
  4.7.1 Preserved Size Selections .......................................................................... 48
  4.7.2 Over Production and Survival Constrained Selections ............................. 49

CHAPTER 5 RESULTS AND ANALYSES ...............................................................51
  5.1 Single Ground Station Scenario ..................................................................... 51
    5.1.1 Schedule for 5 Spacecrafts ...................................................................... 51
    5.1.2 Schedule for 7 Spacecrafts ...................................................................... 54
  5.2 Multiple Ground Stations Scenario ................................................................. 57
  5.3 GA performances ............................................................................................ 60
    5.3.1 Population size ......................................................................................... 60
    5.3.2 Crossover points .................................................................................... 62
    5.3.3 Mutation rate .......................................................................................... 63

CHAPTER 6 CONCLUSIONS AND RECOMMENDATION...................................... 65
  6.1 Conclusions .................................................................................................... 65
  6.2 Recommendations ......................................................................................... 65
  6.3 Future works .................................................................................................. 66

CHAPTER 7 BIBLIOGRAPHY ..................................................................................67

CHAPTER 8 APPENDIX ...........................................................................................69
  8.1 Create the chromosome .................................................................................. 69
  8.2 Crossover operation ....................................................................................... 70
  8.3 Mutation operation ......................................................................................... 71
  8.4 Fitness modules .............................................................................................. 72
    8.4.1 Access Window Fitness calculation ....................................................... 72
    8.4.2 Communication Clashes Fitness calculation ......................................... 74
    8.4.3 Time Requirement Fitness calculation .................................................. 74
    8.4.4 GS usage Fitness calculation ................................................................. 76
  8.5 Codes for a completed single cycle ............................................................... 76
  8.6 Code for 20 cycles with 500 generations of chromosomes ......................... 79
List of Figures

Figure 1.1 Binary encoding method of GAs .............................................................. 16
Figure 1.2 Crossover operation of GAs ................................................................. 16
Figure 1.3 Mutation operations of GAs ................................................................. 16
Figure 1.4 A standard GA process ...................................................................... 17
Figure 2.1 Illustration of a GS AOS and LOS angles .............................................. 22
Figure 2.2 Locations of ESA ground stations (credit ESA) ..................................... 22
Figure 2.3 Ground tracks of ISS, (Credit: A. Barmettler) .................................... 26
Figure 2.4 Visibility clashes of 2 days 5 spacecrafts ............................................. 28
Figure 3.1 ESA Ground Stations on Earth ............................................................ 31
Figure 3.2 Mission spacecrafts ground tracks ...................................................... 31
Figure 3.3 Report of visibility windows ............................................................... 32
Figure 3.4 Ground station and its satellites visibility windows ............................. 32
Figure 3.5 GS[1,2,3,4] and their accessible time window to spacecrafts ............... 33
Figure 4.1 Decimal Encoding, partially ................................................................. 39
Figure 4.2 Encoding of a solution (Chromosome A only), partially displayed, Binary ................................................................. 40
Figure 4.3 Generating initial population ............................................................... 40
Figure 4.4 Crossover of two chromosomes ........................................................ 41
Figure 4.5 Mutation of the chromosome, Decimal Encoding, Rate 0.3 ................. 42
Figure 4.6 Process of checking a chromosome set’s fitness ................................. 43
Figure 4.7 Illustration of the calculating Access Windows Fitness ....................... 44
Figure 4.8 Illustration of calculating the fitness of Communication Clashes .......... 45
Figure 4.9 Illustration of calculating the GS usage fitness .................................... 47
Figure 4.10 Roulette Wheel Selection ................................................................. 48
Figure 4.11 Stochastic Universal Sampling, illustrated based on Roulette Wheel Selection ................................................................. 48
Figure 4.12 Two different of selection models ..................................................... 50
Figure 5.1 Fitness of all objectives during 500 Generations, Single GS, 5 Spacecraft ................................................................. 51
Figure 5.2 Selection Fitness During 500 Generations, Single GS, 5 Spacecraft ...... 51
Figure 5.3 Generated Schedule, Single GS, 5 Spacecraft ..................................... 52
Figure 5.4 Fitness of all objectives during 500 Generations, Single GS, 7 Spacecraft ................................................................. 54
Figure 5.5 Selection Fitness During 500 Generations, Single GS, 7 Spacecraft ...... 55
**Figure 5.6** Generated Schedule, Single GS, 7 Spacecrafts.................................55
**Figure 5.7** Fitness of all objectives during 500 Generations, Multi GS ................57
**Figure 5.8** Selection Fitness During 500 Generations, Multi GS ..........................58
**Figure 5.9** Generated Schedule, Multi GS ...............................................................58
**Figure 5.10:** Fitness of 500 generations with different population size ..........61
**Figure 5.11:** Computation time of 500 Generation (20 runs of GA) with different population size.................................................................62
**Figure 5.12:** Fitness of 500 generations with different crossover points..........63
**Figure 5.13:** Fitness of 500 generations with different mutation rates ..........64
List of tables

Table 2.1 Example of ESA missions ................................................................. 23
Table 2.2 Input Parameters .............................................................................. 29
Table 2.3 Output Parameters ........................................................................... 29
Table 2.4 Intermediate Parameters ................................................................. 30
Table 3.1 Time Requirements of spacecrafts .................................................. 34
Table 5.1 Result of fitness, Single GS, 5 SCs ................................................... 53
Table 5.2 Generated Schedule, Single GS, 5 SCs ............................................. 53
Table 5.3 Result of fitness, Single GS, 7 SCs ................................................... 55
Table 5.4 Generated Schedule, Single GS, 7 SCs ............................................. 56
Table 5.5 Result of fitness, Multi-GS ............................................................... 59
Table 5.6 Schedule result, GS-SC pairs ......................................................... 59
Table 5.7 Generated Schedule, Multi-GS ....................................................... 59
INTRODUCTION

1. Introduction of this work

Ground stations (GS) are required during spacecraft (S/C) operations to provide communications links between operations teams and the S/C systems. Basic communication types, such as telemetry, telecommand and tracking, are all supported by the same satellite and GS systems. Traditionally, the allocation of ground resources to S/C is a highly constrained problem with sufficiently few solutions that the process is conducted manually: stations are selected for communication support with certain S/C for certain periods.

In cases where there are dedicated ground stations for each mission, or if there is a general surplus of potential ground resources, the manual scheduling of GS resources is straightforward. However, a large network of GSs brings a high cost, and in reality ground resources are shared between many missions, who effectively compete for access within a framework of agreed rules. There are many benefits in using a limited number of GSs and sharing the communication resources between different missions. In order to make the process adaptable to different mission scenarios, an automatic scheduling process is needed to perform these operations and to ‘supervise’ the resource allocation.

The problem falls into the combinatorial optimization area. The optimized solution needs to be found under the assumption that at least one solution exists. In the case that there is no feasible schedule solution, an approximate solution needs to be followed according to a predefined set of rules governing resource priority.

There are a few methods that have been developed [1], such as Mixed-integer Linear Programming model, constraint-based methods, multi-objective optimization with generalized differential evolution, fast branch-and-bound and branch-and-cut algorithms, advanced enumerative schemes, and acceleration by cutting plane techniques, etc.

However, those methods have their limitations in solving efficiently large size instances of the problem. The size of spacecraft schedule problem can increase significantly when number of spacecraft, schedule period, and mission requirements grows. So the computational time and complexity is much higher by using those methods.

In this work we propose to approach the problem with Genetic Algorithms (GAs), which have a wide range of uses already in other scientific and engineering domains, along with other examples of artificial intelligence techniques. The GA is a class of algorithm inspired by Darwin’s theory of evolution, i.e. evolutionary algorithms. It seeks to provide fit and approximate solutions to continuous or combinatorial optimization problems that would otherwise demand high computational resources when tackled with conventional algorithms. Long-term, multi-objective, over-constrained S/C scheduling is thus an excellent application area for exploring the benefits that GA could bring into space operations. [2]
2. Motivation of this work

The author has grown many interests on Genetic Algorithms during the study of the master degree, Master in Aerospace Science and Technology. And also recently he has done some research on the coordination of ESA ground station and their space mission crafts.

ESA tracking station network (ESTRACK) is a worldwide system of ground stations providing links between satellites in orbit and Operations Control Centre at ESOC. The schedule operations have been studied. There are the limitations of currently scheduling system. The most significant one is that human interference and manual organizing of the mission schedules are needed. ESA also have started project on developing new scheduling system. However, the Genetic Algorithms are not included as a traditional method.

The interest of research on Genetic Algorithm for GS-SC scheduling is raised in CTAE (Centre de Tecnologia Aerospacial), an aerospace research center located in Barcelona, where the idea of this thesis was brought out.

Dr. Fatos Xhafa (UPC, Professor), and Dr. Ed Chester (CTAE, Header of R+D) supervised the thesis and its related work.
Chapter 1
Background of GA and Schedule Problem

1.1 Genetic Algorithms

1.1.1 A Brief History of GA

The earliest genetic algorithms appeared in late 1950s, programmed on computer by evolutionary biologists. They were seeking to model aspects of natural evolution. At that time it did not occur to them that this method can be used for solving mathematically modeled problems.

By 1962, researchers such as G.E.P. Box, G.J. Friedman, W.W. Bledsoe and H.J. Bremermann had all independently developed evolution-inspired algorithms for function optimization and machine learning. A more successful development in this area came in 1965, when Ingo Rechenberg, then of the Technical University of Berlin, introduced a technique he called evolution strategy, though it was more similar to hill-climbers than to genetic algorithms. In this technique, there was no population or crossover; one parent was mutated to produce one offspring, and the better of the two was kept and became the parent for the next round of mutation. Later versions introduced the idea of a population. Evolution strategies are still employed today by engineers and scientists. [3]

In 1975, John Holland published the book “Adaptation in Natural and Artificial Systems”. Building on earlier research and papers both by Holland himself and by colleagues at the University of Michigan, this book was the first to systematically and rigorously present the concept of adaptive digital systems using mutation, selection and crossover, simulating processes of biological evolution, as a problem-solving strategy.

1.1.2 What is a Genetic Algorithm

A genetic algorithm (GA) is a kind of algorithm that mimics the evolution of natural biology as method for solving mathematical problems. It can be used to optimize complex problems, when traditional methods are hard to find a solution.

There are usually a few basic steps for each GA method:

1. **Initialization**: It includes setup a mathematical model of the problem; Encode the problem using selected method; Create a starting population contain valid solutions to a problem, which is usually randomly generated.
2. **Fitness**: The setup measures the level of fit of each solution in the population; determine how fit are they according to the designed system environment.

3. **Selection**: The step uses one method or different methods to choose the individuals of a population, which will be used for reproduction of next generation.

4. **Crossover**: It replicates the mating process by crossing over selected individuals. The crossover process exchanges chromosome information between two individuals.

5. **Mutation**: By randomly change a very small part of the chromosome of random selected individuals.

6. **Repeat**: Create the new population and repeat the entire process again.

Those Processes are represented in **Figure 1.4**:
1.1.3 Strengths

1. **Parallel approach**: Unlike most other algorithms, which are serial and explore solution space to a problem in one direction each run time. The most important characteristic of genetic algorithms is that they are intrinsically parallel. They are excellent at overcoming the problem of suboptimal, which usually happens with other kind of algorithms. GA uses multiple offspring, which encodes different legal solutions of the problem in multiple directions at the same run. When the direction turns to be a dead end, they can be eliminated by the fitness, and replaced by new, better solutions, which are generated by the process. This characteristic has a greater chance of finding global optimal solutions.

2. **Reduce number of searches**: Genetic algorithms are well suited for nonlinear problems; especially those that have a huge space of potential solutions. Those problems usually take big amount of time for searching exhaustively, even with high computation power. In the real world problems are mostly those nonlinear and complicated cases.

For example, a genetic algorithm developed jointly by engineers from General Electric and Rensselaer Polytechnic Institute produced a high-performance jet engine turbine design that was three times better than a human-designed configuration and 50% better than a configuration designed by an expert system by successfully navigating a solution space containing more than $10^{387}$ possibilities. Conventional methods for designing such turbines are a central part of engineering projects that
can take up to five years and cost over $2 billion; the genetic algorithm discovered this solution after two days on a typical engineering desktop workstation [4].

3. **Exceed Local Optima**: Genetic algorithms perform well in problems for which the fitness landscape is complex. They perform well when the fitness is discontinuous, changes over come, or has many local optima. The challenges of practical problems become to avoid the local optima. Many search algorithms such as hill climbing and simulated annealing methods can be trapped in those local optima, and cannot discover better solution nearby, then finish the search process, even though there a better optima somewhere else in the solution space.

The key process of GA is crossover, which gives the possibility of escape of the trap in the local optima and starts the searching process from another area of the solution space. With the crossover in place, there are information transform between successful solutions, where individuals have the possibilities of getting the good gene that others have, and involve to a better solution.

4. **Multiple Objectives**: Genetic algorithms are able to optimize the problem with different objectives simultaneously. Real world problems mostly cannot be optimized with just one condition, but have to meet with different requirements. GAs are very good at solving such problems: in particular, their use of parallelism enables them to produce multiple equally good solutions to the same problem, possibly with one candidate solution optimizing one parameter and another candidate optimizing a different one [3]. Fitness functions in GA can be added sequentially or parallel to the optimizing process, in order to make the solution meet different objectives.

5. **Comprehensive**: Genetic algorithms don’t need to know (or don’t know) the problem they are solving. Unlike other methods using a domain-specific information to guide each step and making changes with special attention against the improvements, GAs are “blind watchmakers” [5]. They just need to randomly change small part of the individuals and leave the determination to those fitness functions.

1.1.4 **Limitations**

1. **Premature and slow convergence**: If one of the individual is more fit than most of others in the early generations, it might domain the population too soon, and drives down the diversity of following generations. Then the solution has the danger of falling into a local optimum. [14][21]. And also the process of convergence to the global optimal can be quiet slow sometime depend on the GA parameters been chosen.

2. **Fitness functions and GA parameters**: Fitness functions and parameters need to be carefully designed. A little modification sometimes can lead to dramatic change in result, while a big changed in selection method may not reflect in result at all.
1.2 Resource-Constrained Schedule Problems

1.2.1 Definition

In a general way, resource-constrained scheduling problems is:

Given
- A set of tasks that need to be scheduled,
- A set of resources that can be used for performing tasks,
- A set of constraints need to be satisfied,
- A set of objectives with which to judge the performance of the schedule.

In order to
- Find out the best way to assign the resources to the activities at specific times,
- Which is within the limit of constraints,
- And meet the objectives as well as possible.

The meaning of those terms can be defined as following: [7]

**Tasks:** Tasks have measurable estimates of performance criteria such duration, cost, and resource consumption. They might require a single resource or a set of resources, which might vary over the duration of the tasks. A task may have multiple execution modes. Any task may be executed in more than one manner depending upon which resources are used to complete it.

**Resources:** Resources can be renewable or non-renewable. Renewable resources are available each period without being depleted. And non-renewable resources are depleted as they are used. Availability of each resource to tasks may vary from time to time. This is also part of constraints to the schedule.

**Constraints and Objectives:** They are defined during the problem formulation, have different meaning and requirements. Constraints determine the feasibility of a schedule and must be satisfied, while objectives show the optimality of the schedule and should be satisfied as much as possible. They can be task-based, resource-based or related to result measurement.

The result of a schedule can be “Feasible”, which means it satisfies all constraints. Or it can also be “Optimal”, which means it not only satisfies all constraints but also meets the objectives at the best, at least as good as any other result.

But sometimes, there might be not possible to find a suitable schedule according to constraints. In this case, an “as-best-as-possible” schedule that matches as many as of the constraints as possible will be given. So in order to reach to a “Feasible” result, either an increase resources or reduction of constraints has to be done to the schedule process.
1.2.2 Why it is hard to schedule

Scale of the Schedule: Scheduling problem consist of the question what must be done where and when, which means task operates on the resource at specific time. By using this classification, an estimate of problem size can be made. The choice of the representation controls the size of the search space. Very general representations can applied to multiple problems but require a large search space, while the others may significantly reduce the size of search but limit the problems they can solve.

Modeling of Real Problems: Mathematical solutions are always an ideal situation of the real problem, even the most special designed ones. The problem in realities may have more complicated constraints and objectives. Also random error, human introduced malfunction, and emergent condition changes can cause the scheduling break down.

Sparseness of Solution Space: Depending on the representation and modeling assumption, there might be no feasible solution to a problem. Resources can be over used for one task, while another can't complete. Or constraints in the model can never be fully satisfied all together. Or there is not enough resource at all. Some algorithms are able to determine the problem if those these conditions exist. However, most heuristic methods cannot fulfill this.
Chapter 2
Ground Stations, Spacecrafts and Mathematical Modeling

2.1 Ground Stations

2.1.1 What is a Ground station

Ground station is also called earth station. It is a terrestrial terminal designed for extra-planetary communications with spacecraft. Ground stations communicate with a spacecraft by transmitting and receiving radio waves in high frequency bands (e.g. microwaves).

A ground station usually contains more than one satellite dish. Each dish is usually assigned to a specific space mission. With the scheduling from control center, those dishes are able to handle and switch among mission spacecrafts. The research of this project is to find a method of intelligently handling the communication schedule with target spacecrafts.

For the simplification of the project, a Ground Station (GS) defined in this work actually refers to a dish in the real world. In this case a real world ground station is seen as a set of individual GS.

2.1.2 The Visibility of a ground station

There are three types of visibility of a S/C to GS.

- AOS-VIS: Acquisition of Signal, Visible. This indicates the time when the S/C appears in the line of sight of the GS.
- AOS-TM: Acquisition of Signal, Telemetry. This is time when G/S can start receiving telemetry signals from the S/C.
- AOS-TC: Acquisition of Signal, Telecommand. This is time when GS are allowed to send signal to S/C.

The angle of each those are defined with the regulations of GS, so that the high amount of transmitting energy will not harm the organisms on the ground. Similarly defined, we also have Loss of Signals, i.e. LOS-TC, LOS-TM, LOS-VIS.

In this project we are scheduling the events ranging from $T_{AOS-TC}$ to $T_{LOS-TC}$. In the simulation process, to get this data, we will set the elevation angle to 10 degrees for all the spacecraft to simplify the process.

Following Figure 2.1 shows the relations among all those angles.
2.1.3 The Distribution of Ground Stations

In order to cover all the controlled space mission, a network of ground stations are usually required to satisfy the communication need for all the missions. Mission communications are handled among those GS along the rotation of the earth. For example, Deep Space Network (DSN) has three ground facilities placed approximately 120 degrees apart around the world. Those are: Goldstone, in California’s Mojave Desert; near Madrid, Spain; and near Canberra, Australia.

The ground stations of space agencies are usually spread around the globe as much as possible in order to have a good coverage of the sky. See below Figure 2.2, which indicates the related ground stations for ESA [21].
2.2 Spacecraft

2.2.1 Spacecraft and their missions

Depending on the objective of space missions, the designs of each spacecraft are different. However, what really matters to the scheduling problem are their orbits in space. The type of the orbit determines the shape of ground track, and together with the knowledge of the ground station locations, visibility window for the spacecraft will be found.

Missions we are dealing with are mostly scientific missions. Even though in the test cases, we are not using the real requirements from those missions, the assumptions of the communication links are still based on some real spacecrafts. The following Table 2.1 describes objectives of missions and orbits of those spacecraft, which we will be using in the test case.

Table 2.1 Example of ESA missions

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Mission:**
| The four identical Cluster satellites (FM5 to FM8) research the protective magnetosphere of the Earth that shields us from the continual solar wind. They measure three-dimensional data from the collision of the solar wind with the Earth's magnetic field, which changes over time and the effects on near-Earth space and its atmosphere. |
| **SC Orbit:**
| Highly elliptical orbits. The perigee is around 4 RE and apogee 19.6 RE. Each orbit takes approximately 57 hours to complete. |
| **Spacecraft:**
| ![Cluster II spacecraft](image) |
Mission:

Giove, or Galileo In-Orbit Validation Element, are the satellites built to test the technology in orbit for Galileo positioning system, which is an alternative and complementary system to the US GPS and the Russian GLONASS.

SC Orbits:

Giove-A: Circular orbit in MEO at 23.200 KM
Giove-B: Circular orbit in MEO at 23.200 KM

Spacecraft:
| **INTEGRAL** [18] | **Mission:**
| | The task of INTEGRAL (International Gamma-Ray Astrophysics Laboratory) is to gather some of the most energetic radiation that comes from space. It provides a new insight of the Universe such as black holes, neutron stars, active galactic nuclei and supernovas.
| **SC Orbit:**
| | Highly eccentric 72-hour orbit around the Earth, with perigee at 9000 km, apogee at 153 000 km, and inclination of 51.6°
| **Spacecrafts:**
| | ![Image of INTEGRAL spacecraft](image1.jpg)

| **METEOSAT-6** [19] | **Mission:**
| | The objective of these geostationary meteorological satellites is to ensure providing continuous and reliable meteorological observations from space to a large user community.
| **SC Orbits:**
| | METEOSAT-6: Geostationary orbits 67.5E
| | METEOSAT-7: Geostationary orbits 57.5E
| **Spacecrafts:**
| | ![Image of METEOSAT-6 spacecraft](image2.jpg)
| | ![Image of METEOSAT-7 spacecraft](image3.jpg)
<table>
<thead>
<tr>
<th>XMM-NEWTON [20]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mission:</strong></td>
</tr>
<tr>
<td>Since Earth’s atmosphere blocks out all X-rays, only a telescope in space can detect and study celestial X-ray sources. The XMM-Newton mission will help scientists solve a number of cosmic mysteries, ranging from the enigmatic black holes to the origins of the Universe itself.</td>
</tr>
<tr>
<td><strong>SC Orbit:</strong></td>
</tr>
<tr>
<td>48-hour elliptical orbit around the Earth. Inclined at 40° with a Southern apogee at 114 000 km, the perigee altitude is 7000 km</td>
</tr>
<tr>
<td><strong>Spacecraft:</strong></td>
</tr>
<tr>
<td>![Spacecraft Image]</td>
</tr>
</tbody>
</table>

### 2.2.2 Spacecraft Ground Tracks

Ground track is the path on the surface of the Earth directly below a spacecraft. In other words, it is the projection of the spacecraft’s orbit onto the surface of the Earth. Ground track contains a set of points at which the spacecraft pass directly overhead. It gives a good location reference of its ground station. See following Figure 2.3 for the ground track of International Space Station.

![Ground Tracks of ISS](Credit: A. Barmettler)
2.3 The Complexities of Ground Station Scheduling

2.3.1 Limited Ground Stations

We consider all the ground stations are resources. Constraints come from the number of those ground stations. Usually there are limited amount of ground station compared to the number of the space missions need to be supported.

The examples of space missions we are using belong to ESA. And accordingly, the ground station resources distributed around the world are only less than ten, which need to support the agency’s missions. Those ground stations are [8]:

- Kourou (French Guiana),
- Maspalomas, Villafranca and Cebreros (Spain),
- Redu (Belgium),
- Santa Maria (Portugal),
- Kiruna (Sweden),
- Perth and New Norcia (Australia)
- Malargüe (Argentina; under construction)

2.3.2 Complicated Mission Requirements

Depending on the mission, the time required for links can possibly range from less than an hour to 8 hours per day. Besides the time required, there are also all other kinds of mission operation requirements, which are affecting the scheduling process.

Following is an example of partial requirements for two missions M1 and M2. Those two missions are sharing a same ground station G1. Due to the usage of other spacecrafts, it also involves with another ground station G2 [4]:

1. Lunar occultation periods have to be avoided
2. The minimum pass duration is 3-hours for M1, selected from within the physical station visibility
3. The minimum pass duration is 3-hours for M2, selected from within the physical station visibility
4. The separation of pass should be 24 hours +/- 30 minute
5. The maximum separation of passes should be 27 hours
6. The minimum pass elevation should be 5°
7. M1 and M2 pass should be scheduled within a period of 8 hours. It is expected a ground station reconfigure time between spacecrafts of less than 30 minutes, including the pre-pass test
8. The order M1-M2 or M2-M1 should be retained until a change is requested
9. During some period, due to the load of G1, it may required to support on of M1 and M2 spacecraft from G2. The selection of SC support by G2 should be maintained for the full duration of the further analysis.
From the textual descriptions, those requirements contain link duration, separation time of links, order of the links to different spacecrafts, and rules about involvement with another ground station.

### 2.3.3 Spacecraft Visibility Clash

The most common constraints are the clash of visibility windows caused by multiple spacecrafts to a single ground station [2]. A visibility clash of two spacecraft happens when the AOS time of second spacecraft starts before the LOS time of first one.

In easy words, a ground station is always or most of the time has the possibility to establish communications to more than one spacecrafts. Following is a sample for a two-day visibility for 5 spacecraft (the horizontal axis is the time, and vertical are the SCs).

![Visibility clashes of 2 days 5 spacecrafts](image)

**Figure 2.4** Visibility clashes of 2 days 5 spacecrafts

### 2.4 Mathematical model

Mathematical modeling of the problem is the key for solving it. It is divided into two parts. Firstly, is to model the fact related issues, which includes the fact of ground stations, spacecraft, their visibility, and related parameters.

Secondly, and also a difficult part, is to model all the mission rules. Each mission has its different rules about the downlink time, ground station preference, overlapping with other spacecrafts, un-scheduled events, etc. In order to make the problem solvable, all this rules are to be translated into a time span representation with one or more series of start time and its duration.

Almost all the rules can be modeled in this way of time span. However, it can be periodic or not. GA itself will not be doing this part of work. They need to be preprocessed as an input for the system.
2.4.1 Definitions and Assumptions

To make the system mathematically solvable, we have simplified the scheduling system by defining the following rules:

- Maximum link time of each communication is defined.
- A schedule period of 8 days is generated.
- 5 real mission spacecrafts are considered in the Single-GS scenario. 7 mission spacecrafts are considered in Multi-GS scenario. They are: CLUSTER II-FM5 [SC(1)], GIOVE-A [SC(2)], GIOVE-B [SC(3)], INTEGRAL [SC(4)], METEOSAT-6 [SC(5)], METEOSAT-7 [SC(6)], and XMM [SC(7)].
- 4 ground stations are selected for Multi-GS scenario. In reality, they are ESA ground stations: Cebreros [GS(1)], Kourou[GS(2)], Malargue [GS(3)], and New Norica [GS(4)].
- The relations among those ground stations and spacecrafts are assumed. In reality it is a different configuration.
- Each spacecraft has one communication possibility with ground during one day.
- In multiple GS scenario, each spacecraft has consisted communication ground station for the entire scheduled period. That is the spacecraft will not change the ground station once it is selected.
- Mission requirement starts from a very simple one. That is communication time requirement. Other requirements such as priorities, space mission relations are considered as additional fitness module to the solver.

2.4.2 Mathematical Modeling

Table 2.2 Input Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC {i}</td>
<td>Spacecrafts that need to be scheduled in the system</td>
</tr>
<tr>
<td>GS {g}</td>
<td>Ground station resources in the system</td>
</tr>
<tr>
<td>N\text{days}</td>
<td>Number of days for the scheduling period</td>
</tr>
<tr>
<td>T_{AOS-VIS}(i)(g)</td>
<td>Time when spacecraft is geometrically visible</td>
</tr>
<tr>
<td>T_{LOS-VIS}(i)(g)</td>
<td>Time when spacecraft leaves geometric visibility</td>
</tr>
<tr>
<td>T_{Req}(i)</td>
<td>Link time requirement of each spacecraft</td>
</tr>
</tbody>
</table>

Table 2.3 Output Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{start}(i)(g)</td>
<td>Time when the communication link will start, this is output of the schedule system</td>
</tr>
<tr>
<td>T_{dur}(i)(g)</td>
<td>Duration of communication, which represent the length of time that a spacecraft is communicating with a ground station. Another output from the schedule</td>
</tr>
</tbody>
</table>
Ground Stations Scheduling with Genetic Algorithm

SC-GS(i) | Selection of Ground Station of each spacecraft
Fit_{LessClash} | Fitness level of the objective to minimize link clashes
Fit_{VisWin} | Fitness level of the objective to meet the access window of spacecrafts
Fit_{Req} | Fitness level of the objective to fulfill the mission time requirements
Fit_{GSU} | Fitness level of the objective to maximize the usage of ground stations

Table 2.4 Intermediate Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{AOS-link(i)(g)}</td>
<td>Time when start a 2-way communication link is physically possible</td>
</tr>
<tr>
<td>T_{LOS-link(i)(g)}</td>
<td>Time when a link has to be terminated</td>
</tr>
<tr>
<td>T_{possible(i)(g)}</td>
<td>Possible time for communication, It is the possible length of time for communication, which usually is from AOS-Link to LOS-Link. It is a parameter used in fitness function</td>
</tr>
<tr>
<td>T_{waste(i)(g)}</td>
<td>Wasted time of communication, It is the time that spacecraft within T_{possible}, but has no communication with the visible ground station. It is a parameter used in fitness function too</td>
</tr>
<tr>
<td>T_{total(g)}</td>
<td>Total link time, It is total length of time that one ground station is in communication with its entire visible spacecrafts. It is a parameter used for fitness function</td>
</tr>
<tr>
<td>N_{reconfig(g)}</td>
<td>Number of reconfiguration, It represents the how many times of reconfiguration of a ground station. It is a parameter used for fitness function</td>
</tr>
<tr>
<td>P(i)(g)</td>
<td>Priority of spacecrafts to ground stations, and ground stations to spacecrafts</td>
</tr>
</tbody>
</table>
Chapter 3
Simulation and Data Generation

3.1 Mission simulation using STK

Satellite Tool Kit (STK), is software package from Analytical Graphics, Inc. that allows engineers and scientists to design and develop complex dynamic simulations of real-world problems. It is created to solve problems of Earth-orbiting satellites.

It is a powerful software tool. We are using it to simulate a ground stations an its satellites. Based on ESA ground stations, we create them with exact position on Earth.

![Figure 3.1 ESA Ground Stations on Earth](image1)

Then, based on real ESA space mission, orbits of spacecrafts are added to the scenario. The 2D view of spacecrafts ground tracks helps to relate the mission with ground stations.

![Figure 3.2 Mission spacecrafts ground tracks](image2)
3.2 Data generation using STK

3.2.1 Single ground station scenario

Here we choose a ground station located in the Europe (in reality is the location of ESA Cebreros station). The start of scheduling period is set to be 1st of May in 2010, 00:00. And this time is the minute 0 for the whole process. The duration is 8 days, which is 11520 minutes.

STK generates a report of the time stamp of visibilities of the ground to its entire associated spacecraft. In the following Figure 3.3 we are giving a snapshot of a small part.

![Figure 3.3 Report of visibility windows](image)

At same time, it can also generate the graphic view of the visibility windows in the entire scheduling period. In the following figure, from the bottom to the top, these lines represent the visibility windows of SC[1] to SC[7].

![Figure 3.4 Ground station and its satellites visibility windows](image)
3.2.2 Multiple GS scenario

In this scenario the difference consists in the ground stations. However space mission requirements are not changed. We are adding more ground stations to the system. And each spacecraft can have different communicating stations.

The difference in the data generation part is that we need to create the access data of all spacecrafts for each of the four ground stations we selected. Some of the spacecraft may not visible for one or more ground stations all the time. Here are the access graphs generated by STK:

![Figure 3.5 GS[1,2,3,4] and their accessible time window to spacecrafts](image-url)
3.3 Data Pre-Process

3.3.1 Visibility Windows

From those visibility windows, we can generate the visibility table from the ground station to its related spacecrafts. The table then is converted to a matrix format that can be processed by our computer program. The first column contains the numerical names of those satellites. The second column and third column are the TOS and LOS. Here shows a part of this matrix.

\[
\begin{bmatrix}
SC[1], & T_{AOS}[1], & T_{LOS}[1]_1 \\
SC[1], & T_{AOS}[1], & T_{LOS}[1]_2 \\
& \vdots & \\
SC[1], & T_{AOS}[1], & T_{LOS}[1]_N \\
SC[2], & T_{AOS}[2], & T_{LOS}[2]_1 \\
SC[2], & T_{AOS}[2], & T_{LOS}[2]_2 \\
& \vdots & \\
SC[i], & T_{AOS}[i], & T_{LOS}[i]_1 \\
& \vdots & \\
SC[I], & T_{AOS}[I], & T_{LOS}[I]_N \\
\end{bmatrix}
\]

3.3.2 Time Requirements

To make the test case easy, we simplify the mission requirement. We are considering only one condition from the all the mission requirements, that is the communication link time. This defines how long the spacecraft has to communicate with the ground station in a short period.

<table>
<thead>
<tr>
<th>SC</th>
<th>From (min)</th>
<th>To (min)</th>
<th>Require (min)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2880</td>
<td>60</td>
<td>1 hour / 2 days</td>
</tr>
<tr>
<td>1</td>
<td>2881</td>
<td>5760</td>
<td>60</td>
<td>1 hour / 2 days</td>
</tr>
<tr>
<td>1</td>
<td>5761</td>
<td>8640</td>
<td>60</td>
<td>1 hour / 2 days</td>
</tr>
<tr>
<td>1</td>
<td>8641</td>
<td>12960</td>
<td>60</td>
<td>1 hour / 2 days</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2880</td>
<td>80</td>
<td>80 mins / 2 days</td>
</tr>
<tr>
<td>2</td>
<td>2881</td>
<td>5760</td>
<td>80</td>
<td>80 mins / 2 days</td>
</tr>
<tr>
<td>2</td>
<td>5761</td>
<td>8640</td>
<td>80</td>
<td>80 mins / 2 days</td>
</tr>
<tr>
<td>2</td>
<td>8641</td>
<td>12960</td>
<td>80</td>
<td>80 mins / 2 days</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1440</td>
<td>120</td>
<td>2 hours / day</td>
</tr>
<tr>
<td>3</td>
<td>1441</td>
<td>2880</td>
<td>120</td>
<td>2 hours / day</td>
</tr>
<tr>
<td>3</td>
<td>2881</td>
<td>4320</td>
<td>120</td>
<td>2 hours / day</td>
</tr>
<tr>
<td>3</td>
<td>4321</td>
<td>5760</td>
<td>120</td>
<td>2 hours / day</td>
</tr>
<tr>
<td>3</td>
<td>5761</td>
<td>7200</td>
<td>120</td>
<td>2 hours / day</td>
</tr>
<tr>
<td>3</td>
<td>7201</td>
<td>8640</td>
<td>120</td>
<td>2 hours / day</td>
</tr>
</tbody>
</table>


3.3.3 Benchmark

3.3.3.1 Single-GS - 5SC scenario:

<table>
<thead>
<tr>
<th>GS</th>
<th>SC</th>
<th>Schedule days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cebreros</td>
<td>GIOVE-A</td>
<td>8 days</td>
</tr>
<tr>
<td></td>
<td>GIOVE-B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTEGRAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XMM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>From May 1, 2010</td>
<td></td>
</tr>
</tbody>
</table>
### 3.3.3.2 Single-GS - 7SC scenario:

<table>
<thead>
<tr>
<th>GS</th>
<th>SC</th>
<th>Schedule days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cebreros</td>
<td>CLUSTER_II-FM5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTEGRAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XMM</td>
<td>8 days</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From May 1, 2010</td>
</tr>
</tbody>
</table>

### 3.3.3.3 Multi-GS - 7SC scenario:

<table>
<thead>
<tr>
<th>GS</th>
<th>SC</th>
<th>Schedule days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cebreros</td>
<td>CLUSTER_II-FM5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTEGRAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XMM</td>
<td>8 days</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From May 1, 2010</td>
</tr>
<tr>
<td>Kourou</td>
<td>CLUSTER_II-FM5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTEGRAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>METEOSAT-7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XMM</td>
<td>8 days</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From May 1, 2010</td>
</tr>
<tr>
<td>Malargue</td>
<td>CLUSTER_II-FM5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GIOVE-B</td>
<td>8 days</td>
</tr>
<tr>
<td></td>
<td></td>
<td>From May 1, 2010</td>
</tr>
<tr>
<td>New Norica</td>
<td>8 days From May 1, 2010</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>CLUSTER_II-FM5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIOVE-A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIOVE-B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGRAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>METEOSAT-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>METEOSAT-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simulation and Data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGRAL</td>
<td></td>
</tr>
<tr>
<td>METEOSAT-6</td>
<td></td>
</tr>
<tr>
<td>METEOSAT-7</td>
<td></td>
</tr>
<tr>
<td>XMM</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4
GA Process

4.1 Encoding

Encoding the solution into a computable value is a first step of using GA. The encoding methods are different depend on problems. In this project, the problem solution can be represented into two individual chromosomes. One is spacecrafts timetable, which is called Chromosome A. The other is spacecraft and ground station match table, which is called Chromosome B

\[
\text{Chromosome A:} \quad \text{Chromosome B:} \\
\begin{bmatrix}
SC[1], T_{\text{Start}}, T_{\text{Dur}} \\
SC[2], T_{\text{Start}}, T_{\text{Dur}} \\
\vdots \\
SC[i], T_{\text{Start}}, T_{\text{Dur}}
\end{bmatrix} \\
\begin{bmatrix}
SC[1], GS[g_1] \\
SC[2], GS[g_2] \\
\vdots \\
SC[i], GS[g_i]
\end{bmatrix}
\]

4.1.1 Decimal vector encoding

The first and simple method of encoding the solution is simply use a vector to store the real value of each parameter. In the computing process, this structure can be stored either as a matrix or a vector of decimal values.

In Figure 4.1 shows a snapshot of a part of the encoded vector, Where contains two chromosome A and B. Chromosome is a vector of SC, Start time and Duration. Chromosome B is a vector of SC-GS pair.

The advantages of decimal vector encoding are many. The structure itself is a solution for the problem. It is easy to store the values in a matrix for computing. And while applying the mutation operation, different mutation rate can be applied conveniently to different parameters.

4.1.2 Binary vector encoding

Binary encoding is based on the previously encoding method. But also one step further to encode the decimals into binary number and make each entry into a string of binaries.

Figure 4.1 Decimal Encoding, partially
Since binary is the basic compute language, it is easy to operate the GA operations based on the binary encoding. For example, it makes the mutation process easier than previous encoding methods. However, the process of applying differentiated mutation rate is difficult. Also, the conversion of decimal to binary consumes more compute resource than the previous method.

The mutation applied on the binary encoding is based on bit inversion. That is 0 to 1, or 1 to 0. The method is simple. When the mutation is applied, it is hard to know if the mutation is within the designed range.

Based on the advantages of decimal vector encoding, we have chosen this method as the encoding method in the entire following test.

### 4.2 Initial population

A population generator randomly generates the initial population. The generator creates chromosome with input data of population size, number of spacecrafts, ground station and schedule periods. See Figure 4.3.
4.3 Crossover for Reproduction

Multi-points crossover has been selected for solving the problem, because of the length of chromosome we are dealing is usually big. E.g. an 8-Day, 7-SC, scheduling can have a vector chromosome size of 56.

The crossover rate is a genetic parameter for the problem, which is tunable. The crossover points are generated randomly for each crossover operation. Following figure show the process of crossing over two chromosomes.

![Figure 4.4 Crossover of two chromosomes](image)

4.4 Mutation

Mutation is one of the most important GA operators in the entire process. It brings genetic diversity for each generation. It simulates biological mutation.

The mutation operator performs an arbitrary bit change in the chromosome with a certain probability, which is called mutation rate. The higher mutation rate, the more change will be brought to the chromosome. A common way of mutating a chromosome in binary encoding is bit inversion. Bits are changed from 0 to 1, or 1 to 0, according to the mutation probability. However, in the problem case defined in this work we also approach the mutation from another way.

Some of the bit cannot be mutated, such as SC\[i\]. Other bits need to be mutated with an identified range, such as T\(_{\text{Start}}\) and T\(_{\text{Dur}}\). Following Figure 4.5 shows the mutation process of an example chromosome.
4.5 Feasibility of the offspring solutions

When the crossover and mutation produces the offspring, we need to maintain their feasibility to the solution. That is also to guarantee that the new solutions are still a logical solution. This objective has been archived from two parts: the structure design of solutions coding, and mutation method.

While encoding the solutions, we have the spacecraft name $SC(i)$ encoded as the beginning of each entry. And mutation process does not mutation any information of the spacecraft name information. So we can maintain the same amount of scheduled events for the each spacecraft. This is the first important step to keep the feasibility of offspring.

The second step is about those time tags ($T_{\text{start}}$ and $T_{\text{Dur}}$). In each mutation process, those numbers should not be changed too much into an unrealistic number. So the following regulation is applied:

$$
T_{\text{Start}} \subseteq [N_D \times 1440, \ (N_D + 1) \times 1440]
$$
$$
T_{\text{Dur}} \subseteq [T_{\text{Dur}}(\text{Min}), \ T_{\text{Dur}}(\text{Max})]
$$

In the equations above, $N_D$ is the number days where the $SC[i]$ is in the chromosome. And $T_{\text{Dur}}(\text{Min})$ and $T_{\text{Dur}}(\text{Max})$ are minimum required link time and Maximum link time.

Until now we have been able to control the crossover and mutation operator, so that all the offspring are logical solution to the problem.
4.6 Fitness, multi-objectives

Fitness is the key for GA to solving any problem. In this project, the fitness is not defined just by one function, but by multiple objectives, such as: spacecraft constraints, ground station constraints, and mission requirements, etc. The best solution needs to be optimized according to all of those fitness conditions.

Based on the original definitions and assumptions of the problem, different fitness objectives are modulated and applied to solutions. These basic fitness objectives are called Fitness Modules (FMs). The system is designed in a way that FMs are pluggable. So it is convenient to attach additional FMs to the system individually.

![Figure 4.6 Process of checking a chromosome set’s fitness](image)

According to the process of solution fitness check. There are two types of FM, one is Serial FM (yellow color in the following graph), and the other is Parallel FM (shows green color). Total fitness is the function of all the partial fitness. Defined in the following equation. \( \text{Fit}_S(i) \) and \( \text{Fit}_P(j) \) are the fitness values calculated by Serial-FMs or Parallel-FMs.

\[
\text{Fit} = \sum_{i=1}^{n} w_i \times \text{Fit}_S(i) + \sum_{j=1}^{m} w_j \times \text{Fit}_P(j)
\]

Where \( w_i, w_j \) are the weights of FMs, \( \text{Fit}_S(i) \) and \( \text{Fit}_P(j) \) are the fitness values from Serial-FMs and Parallel-FMs, and \( n, m \) are the number of FMs in the system.

In a simple case is the summary of those partial fitness values:

\[
\text{Fit} = \sum_{i=1}^{n} \text{Fit}_S(i) + \sum_{j=1}^{m} \text{Fit}_P(j)
\]
While the fitness checking process started, we make a copy the chromosome set need to be processed. Then apply the process in the following chart. Serial FMs are the FMs that modify the gene entries in chromosomes. We created this feature because those can remove useless gene entries. Apply to the real schedule case is those time entry that is not possible for the link. Parallel FMs are those only checks the fitness of the modified chromosome, but do not change the gene entries of them.

### 4.6.1 Objective 1 - Access Windows

Access windows are the periods of time when spacecrafts are visible and it is possible to setup a communication link with the ground stations. They are computed by STK simulation process and graphically displayed in previous sessions.

The objective is to let generated communication links all (or as many as they can) fall into those access windows. That is to make as many communications as possible.

In the following equation, \( W_{(g,i)} \) is the Access Window set for Ground Station \( g \) and Spacecraft \( i \). \( T_{Start}(s) \) and \( T_{End}(s) \) are the start and end of each access window.

\[
AW(g,i) = \bigcup_{s=1}^{S} \left[ T_{AOS(g,i)}(s), T_{LOS(g,i)}(s) \right]
\]

The final Access Window fitness of the chromosome (\( \text{Fit}_{AW} \)) is calculated as following:

\[
f(n) = \begin{cases} 
1, & \left[ T_{Start}(n), T_{Start}(n) + T_{Dur}(n) \right] \cap W(n_g,n_i) \\
0, & \text{else}
\end{cases}
\]

\[
\text{Fit}_{AW} = \sum_{n=1}^{N} f(n)
\]

In above equations, \( f(n) \) is a single fitness of entry in chromosome \( A \); \( N \) is the length of the chromosome. The final fitness \( \text{Fit}_{AW} \) is the sum of all its entry fitness values. An illustration below shows the algorithm.

![Figure 4.7 Illustration of the calculating Access Windows Fitness](image-url)
4.6.2 Objective 2 - Communication Clashes

A communication clash represents one communication link starts before the end of another communication link [12]. To process this fitness, chromosome entries are sorted by their start time. In sorted chromosome, it satisfy this following function, where n is from 1 to N as the sorted index:

\[ T_{\text{Start}}(n + 1) < T_{\text{Start}}(n) + T_{\text{Dur}}(n) \]

When this occurs, the fitness will be reduced, and one of the clashed entries has to be removed from the chromosome for further calculation. The total fitness of communication clashes is calculate as following:

\[
f(n) = \begin{cases} 
-1, & T_{\text{Start}}(n + 1) < T_{\text{Start}}(n) + T_{\text{Dur}}(n) \\
0, & T_{\text{Start}}(n + 1) \geq T_{\text{Start}}(n) + T_{\text{Dur}}(n) 
\end{cases}
\]

\[
\text{Fit}_{CS} = N + \sum_{n=1}^{N} f(n)
\]

In the above equations, \( N \) is the number of entries in chromosome (scheduled events). And it is the maximum \( \text{Fit}_{CS} \) an chromosome can get for communication clash, which means there are no clash for all the scheduled events. Following is a graph shows the algorithm.

![Figure 4.8 Illustration of calculating the fitness of Communication Clashes](image)

4.6.3 Objective 3 - Time Requirement of Communications
Time requirement is the most basic requirement of the spacecraft communicating with ground facilities. A sufficient amount of time should be guaranteed for TTC (Telemetry, Tracking and Command), also, depend on the mission type, there are different amount of data need to be downloaded from the spacecraft.

For example, satellites that need to download huge amount of image data requires more time for linking with ground station. Some other scientific mission (such Mars Express) may need even more time to communication because the link capacity is less than usual near earth spacecraft.

Those communications, especial for data download tasks are usually a periodical process. And provided as requirements like: 2 hours communication for SC1 each day, 5 hours data downlink for SC2 every 2 days, etc. Scheduling system need this information to be converted into a machine understandable format, and then process them.

A matrix is used for define those requirement. It is the input for the scheduling system. The time requirements are shown as left.

<table>
<thead>
<tr>
<th>SC</th>
<th>From</th>
<th>To</th>
<th>TREQ [Min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC[1]</td>
<td>T_{From}[1]</td>
<td>T_{TO}[1]</td>
<td>T_{REQ}[1]</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC[N]</td>
<td>T_{From}[N]</td>
<td>T_{TO}[N]</td>
<td>T_{REQ}[N]</td>
</tr>
</tbody>
</table>

The fitness is calculated by summarizing all the communication link durations of each spacecraft, and divides them in the required period to compare if the scheduled time matches requirements. Following equations describes the mathematical process.

For \( m \) and \( n \) in following conditions:

\[
T_{\text{Start}}(m) > T_{\text{From}}(k) \\
T_{\text{Start}}(n) + T_{\text{Dur}}(n) < T_{\text{To}}(k)
\]

We have the communication time with the period \( T_{\text{From}}(k) \) and \( T_{\text{To}}(k) \):

\[
T_{\text{Comm}}(k) = \sum_{j=m}^{n} T_{\text{Dur}}(j)
\]

Then we have the Fitness calculated:

\[
f(k) = \begin{cases} 
1, & T_{\text{Comm}}(k) \geq T_{\text{REQ}}(k) \\
0, & \text{else} 
\end{cases}
\]

\[
\text{Fit}_{TR} = \sum_{k=1}^{K} f(k)
\]
The fitness of time requirement is one of the very key elements to evaluate how good the generated schedule is. It is directly related with the mission requirements.

4.6.4 Objective 4 - Maximize GS Usage

Usually, in a GS and SC communication, we want to use the visibility window as much as possible. That is to maximize the usage of those ground station. And try to avoid the idle time of a ground station.

The following Figure 4.9 shows how different schedules affecting the fitness of ground station usage. In two schedules, events have shorter $T_{Dur}$ on the right than the one on the left. And the fitness values show the level of GS usage by each schedule.

![Illustration of calculating the GS usage fitness](image)

So the fitness $Fit_{GU}$ has been design to fulfill this objective. This fitness value is calculated as the percentage of ground stations occupied time by the total amount of the possible communication time. The more a GS is used, the better this schedule is.

In the following formula, $N$ is the number of entries in a chromosome (events of a entire schedule), $G$ is the number of ground stations. $T_{Total}(g)$ is the total available time the ground station

$$Fit_{GU} = \frac{\sum_{n=1}^{N} T_{Dur}(n)}{\sum_{g=1}^{G} T_{Total}(g)} \times 100$$

4.7 Selection Method

Selection method defines the method used for indentifying individuals in the population for crossover operation in order to reproduction.

Traditionally, method like Fitness Proportionate Selection, Tournament Selection, Stochastic Universal Sampling, etc are used for this process. All those are classified
as preserved size selection in the following. Also another model to explore more possibility in crossover combination is described as Over Production and Survival Constrained Selection.

### 4.7.1 Preserved Size Selections

#### 4.7.1.1 Fitness Proportionate Selection

This method is also called Roulette Wheel Selection since it has the similar idea as roulette wheel. Each chromosome is assigned a probability of selection $P_i$. The higher the fitness, the more possible it is to be selected. It is calculated as following, the $F_i$ is the fitness, and $N$ is the total number of individuals in a generation.

$$P_i = \frac{F_i}{\sum_{i=1}^{N} F_i}$$

We can see in Figure 4.10, the illustration shows the pie chart of the probabilities. At each run of a selection process, the process picks the chromosomes where the pointer falls. Those selected chromosomes have the chance to crossover and reproduce.

In fitness proportionate selection, individuals with low probability still have the chance to be selected, while the high probability individuals may not.

![Figure 4.10 Roulette Wheel Selection](image1)

![Figure 4.11 Stochastic Universal Sampling, illustrated based on Roulette Wheel Selection](image2)
4.7.1.2 Stochastic Universal Sampling

We can consider this selection method based on Roulette Wheel Selection. At each run of a selection, instead of only one individual is selected, multiple individuals are selected at evenly spaced intervals. Figure 4.11 shows the result of one run of the selection.

We can also describe the selection process in the formulas here, where R is and random result of to place the first pointer, and S is the set of selected individuals.

\[
\begin{align*}
\text{Fit}_{\text{All}} &= \sum_{i=1}^{N} \text{Fit}(i) \\
R &\in \left( 0, \frac{\text{Fit}_{\text{All}}}{N} \right) \\
S &= \bigcup_{n=1}^{N} R + \frac{\text{Fit}_{\text{All}}}{N} (n - 1)
\end{align*}
\]

4.7.1.3 Tournament Selection

Each tournament selection chose a few individuals at random from a generation of chromosome sets. Then it runs a “tournament” among them to chose the winning individual for crossover. More tournament selections run until the number of the population is reached.

Assume the size of the population is \( N \). The tournament size is \( K \) (\( K < N \)). Probability \( P \) indicates the chance of the best in the tournament been selected. Here we have the algorithm:

1. Random select \( K \) number of individuals from the population.
2. Run the tournament by the fitness.
3. The best-fit individual has the probability \( P \) to be selected. The second best fit individual has the probability \( P \times (1-P) \), the \( j \) best fit has \( P \times (1-P)^{j-1} \).
4. Run step 1 until \( N \) rounds is reached

The special case is when \( P=1 \), it become a deterministic selection. The selection process will always put the best-fit individual as the winner.

4.7.2 Over Production and Survival Constrained Selections

This selection model is trying to explore the best fits by combine multiple selection methods together to get the best fitness individuals. The following Figure 4.12 shows the difference with the Preserved Size Selection.
The significant difference is we have a large intermediate population here, where is a few times more number of individuals then the designed population. Another simple selection based on fitness ranking is applied to this intermediate population, remove the less fit individuals, and reduce the population size to the designed value.

![Figure 4.12 Two different of selection models](image)

The advantage is custom selection rules can be applied. For example, we have tried to apply the following rules together to generate a size of $4 \times N$ intermediate population:

1. The parent generation is cloned to the intermediate population
2. Best-fit individual has the possibility to crossover with all the population
3. Run a process of Roulette Wheel Selection
4. Random crossover each two individuals in the parents generation

Best-fit individuals can be found rapidly this method due to the number of the combination has been increased. However, the drawback is that it can slow down the entire GA process by a big computation factor.
5.1 Single Ground Station Scenario

5.1.1 Schedule for 5 Spacecrafts

In Figure 5.1, shows the fitness curves of 4 different objectives within 500 Generation. The data is the result of a 20 times of running the GA over the same initial conditions, i.e. GA parameters.

<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of Population</td>
<td>20</td>
</tr>
<tr>
<td>Num. of Crossover Points</td>
<td>15</td>
</tr>
<tr>
<td>Mutation rate (Start time)</td>
<td>10%</td>
</tr>
<tr>
<td>Mutation rate (Comm. Duration)</td>
<td>10%</td>
</tr>
</tbody>
</table>

In each sub plot, top orange line is all the best fitness of those 20 rounds. Bottom blue line is the worst fitness in 20 rounds, and the middle purple line are the average fitness of the generation of each around.

Figure 5.1 Fitness of all objectives during 500 Generations, Single GS, 5 Spacecraft
Same process, we have the curves of the selection fitness in the following Figure 5.2. The selection fitness is defined as: 
\[ F_{\text{Sel}} = 3 \times F_{\text{REQ}} + F_{\text{GSU}} \]

Choosing one of the best solutions we are able to get the schedule of the entire 8 day period of those spacecrafts. See Figure 5.3, the timeslots of squares are the Access Window of the ground station to those spacecrafts. And those solid bars are actually scheduled communication time for each spacecrafts.

Here we have a summary of the fitness result:
### Table 5.1 Result of Fitness, Single GS, 5 SCs

<table>
<thead>
<tr>
<th>Objective</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less Link Clashed</td>
<td>34</td>
</tr>
<tr>
<td>Fit to Access Windows</td>
<td>33</td>
</tr>
<tr>
<td>Link Time Required</td>
<td>24 [4 4 8 4 4]</td>
</tr>
<tr>
<td>Ground Station Usage</td>
<td>65</td>
</tr>
</tbody>
</table>

### Table 5.2 Generated Schedule, Single GS, 5 SCs

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Link Start</th>
<th>Link End</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>183 2525 4165 6769 8350 11000</td>
<td>214 170 106 168 205 141</td>
</tr>
<tr>
<td>2</td>
<td>437 2760 2910 4612 6943 8694</td>
<td>233 143 270 292 276 307</td>
</tr>
<tr>
<td>3</td>
<td>735 1996 3472 5361 6089 7601 9001 10447</td>
<td>123 268 180 298 297 271 266 295</td>
</tr>
<tr>
<td>4</td>
<td>1191 2345 3973 5046 7926 9323 11142</td>
<td>261 152 141 287 309 184 255</td>
</tr>
<tr>
<td>5</td>
<td>871 1513 3739 4396 7346 9574 10154</td>
<td>185 283 141 186 243 95 259</td>
</tr>
</tbody>
</table>
5.1.2 Schedule for 7 Spacecrafts

We are giving the same GA parameters as the for 5 spacecrafts scenario:

<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of Population</td>
<td>20</td>
</tr>
<tr>
<td>Num. of Crossover Points</td>
<td>15</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>10%</td>
</tr>
</tbody>
</table>

![Figure 5.4](image-url) Fitness of all objectives during 500 Generations, Single GS, 7 Spacecraft
The summary of one of the best results:

**Table 5.3 Result of fitness, Single GS, 7 SCs**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less Link Clashed</td>
<td>38</td>
</tr>
<tr>
<td>Fit to Access Windows</td>
<td>35</td>
</tr>
<tr>
<td>Link Time Required</td>
<td>33</td>
</tr>
<tr>
<td>Ground Station Usage</td>
<td>65</td>
</tr>
</tbody>
</table>

Figure 5.5 Selection Fitness During 500 Generations, Single GS, 7 Spacecraft

Figure 5.6 Generated Schedule, Single GS, 7 Spacecraft
Table 5.4 Generated Schedule, Single GS, 7 SCs

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Link Start</th>
<th>Link End</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1915</td>
<td>209</td>
</tr>
<tr>
<td>1</td>
<td>7806</td>
<td>133</td>
</tr>
<tr>
<td>1</td>
<td>10540</td>
<td>187</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>205</td>
<td>128</td>
</tr>
<tr>
<td>2</td>
<td>4255</td>
<td>157</td>
</tr>
<tr>
<td>2</td>
<td>6127</td>
<td>232</td>
</tr>
<tr>
<td>2</td>
<td>10192</td>
<td>300</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>637</td>
<td>108</td>
</tr>
<tr>
<td>3</td>
<td>4821</td>
<td>97</td>
</tr>
<tr>
<td>3</td>
<td>7159</td>
<td>105</td>
</tr>
<tr>
<td>3</td>
<td>8692</td>
<td>193</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1033</td>
<td>256</td>
</tr>
<tr>
<td>4</td>
<td>2338</td>
<td>196</td>
</tr>
<tr>
<td>4</td>
<td>3283</td>
<td>256</td>
</tr>
<tr>
<td>4</td>
<td>5592</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>6546</td>
<td>274</td>
</tr>
<tr>
<td>4</td>
<td>8371</td>
<td>227</td>
</tr>
<tr>
<td>4</td>
<td>9289</td>
<td>143</td>
</tr>
<tr>
<td>4</td>
<td>11006</td>
<td>97</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>908</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>2231</td>
<td>73</td>
</tr>
<tr>
<td>5</td>
<td>3619</td>
<td>178</td>
</tr>
<tr>
<td>5</td>
<td>5082</td>
<td>154</td>
</tr>
<tr>
<td>5</td>
<td>10842</td>
<td>137</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>753</td>
<td>152</td>
</tr>
<tr>
<td>6</td>
<td>2671</td>
<td>118</td>
</tr>
<tr>
<td>6</td>
<td>3831</td>
<td>286</td>
</tr>
<tr>
<td>6</td>
<td>5285</td>
<td>187</td>
</tr>
<tr>
<td>6</td>
<td>6870</td>
<td>254</td>
</tr>
<tr>
<td>6</td>
<td>7948</td>
<td>153</td>
</tr>
<tr>
<td>6</td>
<td>9683</td>
<td>124</td>
</tr>
<tr>
<td>6</td>
<td>11183</td>
<td>231</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1672</td>
<td>171</td>
</tr>
<tr>
<td>7</td>
<td>4438</td>
<td>198</td>
</tr>
<tr>
<td>7</td>
<td>7359</td>
<td>141</td>
</tr>
</tbody>
</table>
5.2 Multiple Ground Stations Scenario

In this Scenario, we only take into account 7 spacecraft with 4 ground stations. Given the GA parameters:

<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of Pop.</td>
<td>20</td>
</tr>
<tr>
<td>Num. of Crossover Points (Chromosome A)</td>
<td>10</td>
</tr>
<tr>
<td>Num. of Crossover Points (Chromosome B)</td>
<td>3</td>
</tr>
<tr>
<td>Mutation rate (Chromosome A)</td>
<td>10%</td>
</tr>
<tr>
<td>Mutation rate (Chromosome B)</td>
<td>40%</td>
</tr>
</tbody>
</table>

The fitness for different objectives are shown as following

Figure 5.7 Fitness of all objectives during 500 Generations, Multi GS
Figure 5.8 Selection Fitness During 500 Generations, Multi GS

Figure 5.9 Generated Schedule, Multi GS
The summary information of the schedule generated:

**Table 5.5 Result of fitness, Multi-GS**

<table>
<thead>
<tr>
<th>Objective</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less Link Clashed</td>
<td>46</td>
</tr>
<tr>
<td>Fit to Access Windows</td>
<td>55</td>
</tr>
<tr>
<td>Link Time Required</td>
<td>35 [4 3 4 8 8 4 4]</td>
</tr>
<tr>
<td>Ground Station Usage</td>
<td>30</td>
</tr>
</tbody>
</table>

**Table 5.6 Schedule result, GS-SC pairs**

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Ground Station</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 5.7 Generated Schedule, Multi-GS**

<table>
<thead>
<tr>
<th>Spacecraft</th>
<th>Link Start</th>
<th>Link End</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>308</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>2259</td>
<td>361</td>
</tr>
<tr>
<td></td>
<td>3514</td>
<td>216</td>
</tr>
<tr>
<td></td>
<td>5068</td>
<td>410</td>
</tr>
<tr>
<td></td>
<td>6159</td>
<td>335</td>
</tr>
<tr>
<td></td>
<td>8258</td>
<td>161</td>
</tr>
<tr>
<td></td>
<td>10858</td>
<td>404</td>
</tr>
<tr>
<td>2</td>
<td>1656</td>
<td>335</td>
</tr>
<tr>
<td></td>
<td>6025</td>
<td>404</td>
</tr>
<tr>
<td></td>
<td>7903</td>
<td>353</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>342</td>
</tr>
<tr>
<td></td>
<td>2531</td>
<td>269</td>
</tr>
<tr>
<td></td>
<td>4477</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>6388</td>
<td>343</td>
</tr>
<tr>
<td></td>
<td>10318</td>
<td>385</td>
</tr>
</tbody>
</table>
5.3 GA performances

Changes of GA parameters can influence the convergence of fitness. To explore a set of optimized parameters. We are testing the GA with different parameters, such as population size, crossover points, and mutation rates.

5.3.1 Population size

Run GA with population sizes of 4, 8, 12, 16, 20 and 24. At each population size, the GA is run 20 times. Average fitness values in each generations are calculated in end. Also the computing time cost is recorded. See following.
<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of Crossover Points (Chromosome A)</td>
<td>10</td>
</tr>
<tr>
<td>Num. of Crossover Points (Chromosome B)</td>
<td>3</td>
</tr>
<tr>
<td>Mutation rate (Chromosome A)</td>
<td>10%</td>
</tr>
<tr>
<td>Mutation rate (Chromosome B)</td>
<td>40%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computer</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>2.26GHz Intel Core 2 Duo</td>
</tr>
<tr>
<td>RAM</td>
<td>2G, 1067MHz DDR3</td>
</tr>
<tr>
<td>Operation System</td>
<td>Mac OS 10.6.4</td>
</tr>
<tr>
<td>Software</td>
<td>Matlab 2009b</td>
</tr>
</tbody>
</table>

Figure 5.10: Fitness of 500 generations with different population size
From the first graph we can see that, the more individuals in the population, the better result we can have, however, the second graph shows that there is an big computation increase when the size grows.

So, by comparing the result and computational time, we suggested that population size between 12 and 16 are good values for this GA.

5.3.2 Crossover points

The Figure 5.12 shows the results by using different crossover points, with the single GS scenario.

The parameters used are:

<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of population size</td>
<td>12</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>10%</td>
</tr>
</tbody>
</table>
What we can see from here is that a change in crossover points is not affecting the GA performance that much. This is mainly due to the catachrestic of crossovers. This operator simply exchanges parts of chromosome among individuals, however it does not bring new parts for the chromosomes in the GA. And what changes the big picture a lot is the mutation.

5.3.3 Mutation rate

Base on the same scenario, we study the changed of fitness based on different mutation rates. The result is shown in Figure 5.13.

The parameters used are:

<table>
<thead>
<tr>
<th>GA Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. of population size</td>
<td>12</td>
</tr>
<tr>
<td>Num. of crossover points</td>
<td>10</td>
</tr>
</tbody>
</table>
From here, we can see the performance of GA difference a lot while change the mutation rate. In the figure, it shows that the smaller mutation is, the better result it will get. The means too much changes in the chromosomes brings too many uncertainties, which is not good for the populations evolution.
Chapter 6
Conclusions and Recommendation

6.1 Conclusions

We have defined two scenarios of the GS-SC scheduling problem. One is Single-GS, and the other is Multi-GS. They are based on the same mathematical model with slightly different chromosome structure and fitness checking methods. However, the Single-GS can be considered as a special case of Multi-GS scenario, which can be solved with a general schema.

Two different encoding methods are introduced and tested. There are binary vector encoding and decimal vector encoding. It has been proved that the decimal encoding has a better performance and easy fitness process than the binary encoding. And it is selected as an encoding method for solving the problem.

One important method of solving the problem is the multiple fitness modules in the GA. They implement the objectives for different requirements. Even these fitness modules defined in this paper are not yet sophisticated. But with the use of modularization, it is possible to add and remove mission objectives independently without affecting the rest of the system.

Apart from the traditional selection method, we have designed a combinational selection method of Over Produced and Survival Constrained Selection. It uses a small part of additional computation power but increases the total fitness climbing speed.

Using this designated GA algorithm, we are able to solve the schedule problem between ground stations and spacecrafts. The entire process are simulated and executed in Matlab environment, with the input data generated by STK software. And the results satisfied our initial ideas about this problem.

6.2 Recommendations

This work is focusing on Genetic Algorithms to optimize schedule problem. However, GA is a powerful tool that can be found in lot kinds of optimization processed.

We designed this GA specific GA model for solving schedule problem of ground stations and their spacecrafts. After running tests and analyzing results, we are recommending following parameters for this GA:

- **Encoding:** Two chromosomes structure (represent SC communication time and SC-GS selection pair) is used to encode the solution
- **Crossover rate:** In this GA, this parameter does not effect the final performance that much. However, a relatively high number can be selected (around 60%).
- **Mutation rate:** We suggest keeping a small mutation rate, range from 0.5% to 5% is a good one.
- **Selection Method:** Different methods are encouraged to execute the GA.
- **Population size:** Number of individuals between 12 and 16 are good for achieving nice results, at the same time does not requires too much computational power.

The entire GA automation schedule process reduced human interventions efficiently for schedule problem. In general, Genetic Algorithms are recommended as a (or at least, a strong alternative) method where ground stations and spacecraft schedule is needed.

### 6.3 Future works

The GA we have designed has two major weaknesses:

1. It is not flexible of handling communication handovers. For now, once the spacecraft selects its ground station for communication. This relationship last until the end of the schedule period. It is not possible to use more than one ground resource for the same mission.

2. Number of maximum communications per day need to be predefined. So when handling non-periodical task, this GA doesn’t have a good performance.

However, the future works can be in these two directions based on previous weaknesses:

1. To define a more sophisticated mathematical model that allows the operation of ground station handover. In can be an additional input data for the schedule system that can be encoding as another chromosome of the GA.

2. In order to schedule non-periodical tasks, studies and implementation of variable length chromosome can be included. However, it will change the method of the crossover and mutation, and related fitness modules too.
Chapter 7
Bibliography


Chapter 8
Appendix

8.1 Create the chromosome

```matlab
function eCell = cellCreator(nDays, nSats, nGS, tDur_min, tDur_max)

% use a random generated matrix as a chromosome
% Ts: random {0 - 1440} x Nd
% Tdur: random {0 - 400} // maximum duration can be changed

%--------------------------------------
% process the 1st chromosome, chroA
% it the Sats time schedule table
%--------------------------------------
 nRow = nDays*nSats;
 tStart = myrandint(nRow,1,[1:1440]);
 tDur = myrandint(nRow,1,[tDur_min:tDur_max]);
 chroA = zeros(nRow, 3);

% first row, number of sats
 for m = 1:nDays
  for n = 1:nSats
   chroA((m-1)*nSats+n, 1) = n;
  end
 end

% second row, start time
 for g=1:nDays
  chroA(((g-1)*nSats+1:g*nSats), 2) = tStart(((g-1)*nSats+1:g*nSats), 1)+(g-1)*1440;
 end

% third row, duration
 chroA(:,3) = tDur(:, 1);

%--------------------------------------
% process the 2nd chromosome, chroB
% it the Sats - GS table
%--------------------------------------
 chroB = zeros (n,2);

 for n = 1:nSats
  chroB(n,1) = n;
  chroB(n,2) = randi(nGS);
 end

%--------------------------------------
% put chroA and chroB in a cell
```

8.2 Crossover operation

% function to crossover two chromosomes
% n_points defines how many crossover points

function [new_eCell_1, new_eCell_2] = cellCrossover(eCell_1, eCell_2, a_points, b_points)

chro1A = eCell_1{1};
chro1B = eCell_1{2};
chro2A = eCell_2{1};
chro2B = eCell_2{2};
new_eCell_1 = cell(size(eCell_1));
new_eCell_2 = cell(size(eCell_2));

%---------------------------------------------
% process the ChroA of eCell1 and eCell2
%---------------------------------------------
chroA_length = size(chro1A,1);
points = sort(randperm(chroA_length, a_points));
temp = zeros(size(chro1A));
new_chro1A = chro1A;
new_chro2A = chro2A;
for m = 1:a_points
    temp(points(m):chroA_length,:) = new_chro1A(points(m):chroA_length,:);
    new_chro1A(points(m):chroA_length,:) = new_chro2A(points(m):chroA_length,:);
    new_chro2A(points(m):chroA_length,:) = temp(points(m):chroA_length,:);
end

%---------------------------------------------
% process the ChroB of eCell1 and eCell2
%---------------------------------------------
chroB_length = size(chro1B,1);
points = sort(randperm(chroB_length, b_points));
temp = zeros(size(chroB));
new_chro1B = chro1B;
new_chro2B = chro2B;

for m = 1:b_points
    temp(points(m):chroB_length,:) =
    new_chro1B(points(m):chroB_length,:);
    new_chro1B(points(m):chroB_length,:) =
    new_chro2B(points(m):chroB_length,:);
    new_chro2B(points(m):chroB_length,:) =
    temp(points(m):chroB_length,:);
end

%---------------------------------------------
% Form the new eCells
%---------------------------------------------
new_eCell_1{1} = new_chro1A;
new_eCell_1{2} = new_chro1B;
new_eCell_2{1} = new_chro2A;
new_eCell_2{2} = new_chro2B;

end

8.3 Mutation operation

function eCell = cellMutate(old_eCell, m_rate_a1, m_rate_a2,
m_rate_b, tDur_min, tDur_max)

% Function for mutating a chromosome set at a random rate
% tDur_min, and tDur_max are the retractions of mutation range

old_chroA = old_eCell{1};
old_chroB = old_eCell{2};
eCell = cell(1,2);

%-------------------ChroA-------------------
chroA_length = size(old_chroA,1);
chroA = old_chroA;
ral1 = randperm(chroA_length);
n_TsMu = floor(m_rate_a1*chroA_length);
ra2 = randperm(chroA_length);
n_TdurMu = floor(m_rate_a2*chroA_length);
% mutate Tstart by adding a random number of minute less than 1440 (a day). If the time exceeds a day, then add the rem to the current day
for m = 1:n_TsMu
    Ts = old_chroA(ra1(m),2);
    this_day = floor(Ts/1440);
    Ts_new = Ts + randi(1440);
    if (floor(Ts_new/1440)>this_day)
        Ts_new = this_day*1440 + rem(Ts_new, 1440);
    end
    chroA(ra1(m),2) = Ts_new;
end

% similar idea to the Tstart
for m = 1:n_TdurMu
    Tdur = old_chroA(ra2(m),3);
    Tdur_new = Tdur + randi(tDur_max);
    if (Tdur_new > tDur_max)
        Tdur_new = Tdur_new - tDur_max + tDur_min;
    end
    chroA(ra2(m),3) = Tdur_new;
end
%-------------------------------------
%---------------ChroB-----------------
chroB = old_chroB;
chroB_length = size(old_chroB,1);

rb = randperm(chroB_length);
n_GsMu = floor(m_rate_b * chroB_length);

for n = 1:n_GsMu
    Gs = chroB(rb(n),2);
    Gs = Gs + randi(2);
    if Gs > 3
        Gs = Gs - 3;
    end
    chroB(rb(n),2) = Gs;
end
%-------------------------------------
eCell{1} = chroA;
eCell{2} = chroB;

8.4 Fitness modules
8.4.1 Access Window Fitness calculation
function [f_timeWin newCell] = fitTimeWin(satsVisCell, eCell)
% This function is to calculate the fitness of each chromosome by time window visibility.

% initialize the parameters
newCell = cell(size(eCell));

f_timeWin = 0;  %set the initial fit to zero

chroA = eCell{1};
chroB = eCell{2};

chroA_leng = size(chroA,1);
chroB_leng = size(chroB,1);

% start the calculation
for m = 1:chroA_leng
  GS = 0;
  Sat = chroA(m,1);
  for n = 1:chroB_leng
    if chroB(n,1) == Sat;
      GS = chroB(n,2);
      break;
    end
  end
  if GS == 0;
    error('cannot find match GS');
  end
  % above process, checks if the GS conditions are right
  if size(satsVisCell{GS}{Sat},1) == 0
    chroA(m,2) = 0;
    chroA(m,3) = 0;
    continue;
  else
    for k = 1:size(satsVisCell{GS}{Sat},1)
      if chroA(m,2)>=satsVisCell{GS}{Sat}(k,2) &&
      chroA(m,2) + chroA(m,3)<=satsVisCell{GS}{Sat}(k,3)
        f_timeWin = f_timeWin+1;
        break;
      elseif k == size(satsVisCell{GS}{Sat},1)
        chroA(m,2) = 0;
        chroA(m,3) = 0;
      end
    end
  end
end

%Generate a new chromosome set, where invalid entries are moved
newCell{1} = chroA;
newCell{2} = chroB;
end

8.4.2 Communication Clashes Fitness calculation

function [f_clash newCell] = fitLessClash(eCell)

% Function to create the fitness based on less link clash
newCell = cell(size(eCell));

% calculate perfect fit number
f_clash = size(eCell{1},1)-1;

% sort the chromosome by the start time
[x, i] = sort(eCell{1}(:,2));
chroA = eCell{1}(i,:);
chroB = eCell{2};

% calculate the fitness
chroA_leng = size(chroA,1);

% only if the end of the 1st comm is later than 2nd comm, and they are assign to the same GS
for m = 1:chroA_leng-1
    if chroA(m,2)+chroA(m,3) > chroA(m+1,2) &&
        chroB(chroA(m,1),2) == chroB(chroA(m+1,1),2)
        f_clash = f_clash-1;
        chroA(m,2) = 0;
        chroA(m,3) = 0;
    end
end

% generate a new chromosome set, where invalid entries are moved
newCell{1} = chroA;
newCell{2} = chroB;
end

8.4.3 Time Requirement Fitness calculation

function f_req = fitTimeReq(satsTimeReqCell, eCell)

% function of calculating fitness which is from mission requirement of communication time. Sat_Time_Req.csv file is need to generate the satsTimeReqCell before the execution of this function.
% initialize the parameters, satTimeResult cell is created to store the communication time in required sessions
f_req = 0;
chroA = eCell{1};
chroA_leng = size(chroA,1);
satsTimeResult = satsTimeReqCell;
for m = 1:size(satsTimeResult,1)
    satsTimeResult{m}(:,5) = 0;
end

% start to generate values for satTimeResult
for m = 1:chroA_leng
    Sat = chroA(m,1);
    for n = 1:size(satsTimeResult{Sat},1)
        if chroA(m,2)>=satsTimeResult{Sat}(n,2) && chroA(m,2)+chroA(m,3)<=satsTimeResult{Sat}(n,3)
            if satsTimeResult{Sat}(n,2)+satsTimeResult{Sat}(n,5) > chroA(m,2)
                satsTimeResult{Sat}(n,5) = chroA(m,2)+chroA(m,3)-satsTimeResult{Sat}(n,2);
            else
                satsTimeResult{Sat}(n,5) = satsTimeResult{Sat}(n,5) + chroA(m,3);
                satsTimeResult{Sat}(n,5) = satsTimeResult{Sat}(n,5) + chroA(m,3);
                satsTimeResult{Sat}(n,5) = satsTimeResult{Sat}(n,5) + satsTimeResult{Sat}(n,3) - chroA(n,2);
                if k < size(satsTimeResult{Sat},1)
                    satsTimeResult{Sat}(n+1,5) = satsTimeResult{Sat}(n+1,5) + chroA(m,2) + chroA(m,3)-satsTimeResult{Sat}(n,3);
                end
            end
        break;
        elseif chroA(m,2)<satsTimeResult{Sat}(n,2) && chroA(m,2)+chroA(m,3)>satsTimeResult{Sat}(n,3)
            satsTimeResult{Sat}(n,5) = satsTimeResult{Sat}(n,5) + satsTimeResult{Sat}(n,3) - chroA(n,2);
            if k < size(satsTimeResult{Sat},1)
                satsTimeResult{Sat}(n+1,5) = satsTimeResult{Sat}(n+1,5) + chroA(m,2) + chroA(m,3)-satsTimeResult{Sat}(n,3);
            end
        end
    end
end

% calculate the fitness
for m = 1:size(satsTimeResult,1)
    for n = 1:size(satsTimeResult{m},1)
        if satsTimeResult{m}(n,5) >= satsTimeResult{m}(n,4)
            f_req = f_req + 1;
        end
    end
end
8.4.4 GS usage Fitness calculation

```
function f_GsU = fitGsUsage(eCell, nGS, nDays)

% This function to calculate the fitness based on usage of the GS

    t_comm_all = 0;
    chroA = eCell{1};
    for m = 1:size(chroA,1)
        t_comm_all = t_comm_all + chroA(m,3);
    end
    f_GsU = floor(t_comm_all/(nGS*nDays*1440) * 100);
end
```

8.5 Codes for a completed single cycle

```
function [f1Set f2Set f3Set f4Set fSet] = oneRound()

% Initialize conditions
% GA parameters
nPop = 20;
%nNextPop = 2*nPop;
ncrossoverpoints_a = 10;
ncrossoverpoints_b = 3;
rma1 = 0.1;    % mutation rate of start time
rma2 = 0.1;    % mutation rate of duration time
rmb = 0.4;     % mutation rate of GS

% Mission parameters
nsats = 7;
nGS = 4;
nDays = 8;

tDur_min = 30;
tDur_max = 420;

% Produce first generation
initGen = cell(nPop,1);    %first population
initFits = zeros(nPop,1);
    for m = 1:nPop
        initGen{m} = cellCreator(nDays,nsats,nGS,tDur_min,tDur_max);
```
% process input data
Flist = cell(4,1);
Flist{1} = 'Sat_Vis_1.csv';
Flist{2} = 'Sat_Vis_2.csv';
Flist{3} = 'Sat_Vis_3.csv';
Flist{4} = 'Sat_Vis_4.csv';
satsVisCell = procTimeWin(Flist,nSats);

F = 'Sat_Time_Req.csv';
satsTimeReqCell = procTimeReq(F,nSats);

% calculate fitness of first generation
for m = 1:nPop
    \[f1 \text{ tempCell}] = \text{fitLessClash(initGen}\{m\});
    \[f2 \text{ tempCell}] = \text{fitTimeWin(satsVisCell, tempCell)};
    f3 = fitTimeReq(satsTimeReqCell, tempCell);
    f4 = fitGsUsage(tempCell, nGS, nDays);
    initFits(m) = f1+f2+f3+f4;
end

% Start processing the generations

% define the current Generation
crntGen = initGen;
crntFits = initFits;

% define the next Generation
nextGen = cell(4*nPop,1);
nextFits = zeros(4*nPop,1);

\begin{verbatim}
counter = 1;
\end{verbatim}

\begin{verbatim}
\textcolor{red}{approximated:}
while \text{counter} \leq 500
    \[B \text{ IX}] = \text{sort(crntFits, 'descend')};

    \begin{verbatim}
    \text{crnt generation is copy to the selection}
    \text{for } m = \text{ 1:} n\text{Pop}
        nextGen\{m\} = \text{crntGen}\{\text{IX}(m)\};
    \text{end}

    \text{best cell has the chance to mate with all other cells}
    \text{for } m = \text{ 1:} n\text{Pop}
        [nextGen\{n\text{Pop}+m\} nextGen\{2*}n\text{Pop+m\}] \text{ = cellCrossover(...}
        cellMutate(crntGen\{\text{IX}(1)\}, rma1, rma2, rmb, \text{tDur_min, tDur_max}), \text{...}
        cellMutate(crntGen\{\text{IX}(m)\}, rma1, rma2, rmb, \text{tDur_min, tDur_max}),
    \end{verbatim}
\end{verbatim}
...  

n_crossoverpoints_a, n_crossoverpoints_b...

);  

end  

% random mate each two chromosomes  
r_set = randperm(nPop);  

for m = 1:2:nPop  

[nextGen{3*nPop+m} nextGen{3*nPop+m+1}] ...
           = cellCrossover(...  

   cellMutate(crntGen{r_set(m)},rma1,rma2,rmb,tDur_min,tDur_max),...

   cellMutate(crntGen{r_set(m+1)},rma1,rma2,rmb,tDur_min,tDur_max ),...

       n_crossoverpoints_a,n_crossoverpoints_b...

);  

end  

% until here 4 x nPop number of chromosomes after  
crossover  

% Calculating the fitness:  
for m = 1:4*nPop  

[f1 tempCell] = fitLessClash(nextGen{m});  
[f2 tempCell] = fitTimeWin(satsVisCell,tempCell);  
   f3 = fitTimeReq(satsTimeReqCell,tempCell);  
   f4 = fitGsUsage(tempCell,nGS,nDays);  
nextFits(m) = 3*f3+f4;  
end  

[B IX] = sort(nextFits, 'descend');  

% select the nPop number of best cells  
for m = 1:nPop  

   crntGen{m} = nextGen{IX(m)};  
   crntFits(m) = nextFits(IX(m));  
end  

[f1Set(counter) tempCell] = fitLessClash(crntGen{1});
[f2Set(counter) tempCell] =  
   fitTimeWin(satsVisCell,tempCell);
   f3Set(counter) = fitTimeReq(satsTimeReqCell,tempCell);
   f4Set(counter) = fitGsUsage(tempCell,nGS,nDays);
   fSet(counter) = crntFits(1);  

   counter = counter+1;  

end
8.6 Code for 20 cycles with 500 generations of chromosomes

% Set the number of rounds for a completed GA process
nRound = 20;

% initialize the container for result storing
f1Set = cell(nRound,1);
f2Set = cell(nRound,1);
f3Set = cell(nRound,1);
f4Set = cell(nRound,1);
fSet = cell(nRound,1);

% The calculation process
for n=1:nRound
    disp(strcat('Process to: ', num2str(n)));
    [f1Set{n} f2Set{n} f3Set{n} f4Set{n} fSet{n}] = oneRound();
end

% The following blocks of codes are generating results
for m=1:size(f1Set{1},2)
    f_sum=0;
    for k=1:nRound
        f_sum = f_sum + f1Set{k}(m);
    end
    f1Mean(m) = floor(f_sum/nRound);

    f_sum=0;
    for k=1:nRound
        f_sum = f_sum + f2Set{k}(m);
    end
    f2Mean(m) = floor(f_sum/nRound);

    f_sum=0;
    for k=1:nRound
        f_sum = f_sum + f3Set{k}(m);
    end
    f3Mean(m) = floor(f_sum/nRound);

    f_sum=0;
    for k=1:nRound
        f_sum = f_sum + f4Set{k}(m);
    end
    f4Mean(m) = floor(f_sum/nRound);

    f_sum=0;
    for k=1:nRound
        f_sum = f_sum + fSet{k}(m);
    end
    fMean(m) = floor(f_sum/nRound);
end
for m=1:size(f1Set{1},2)
    f_min = 99999;
    for k = 1:nRound
        if f1Set{k}(m)<f_min
            f_min = f1Set{k}(m);
        end
    end
    f1Min(m) = f_min;

    f_min = 99999;
    for k = 1:nRound
        if f2Set{k}(m)<f_min
            f_min = f2Set{k}(m);
        end
    end
    f2Min(m) = f_min;

    f_min = 99999;
    for k = 1:nRound
        if f3Set{k}(m)<f_min
            f_min = f3Set{k}(m);
        end
    end
    f3Min(m) = f_min;

    f_min = 99999;
    for k = 1:nRound
        if f4Set{k}(m)<f_min
            f_min = f4Set{k}(m);
        end
    end
    f4Min(m) = f_min;
end

for m=1:size(f1Set{1},2)
    f_max = 0;
    for k = 1:nRound
        if f1Set{k}(m)>f_max
            f_max = f1Set{k}(m);
        end
    end
    f1Max(m) = f_max;
end

f_max = 0;
for k = 1:nRound
    if f2Set{k}(m) > f_max
        f_max = f2Set{k}(m);
    end
end
f2Max(m) = f_max;

f_max = 0;
for k = 1:nRound
    if f3Set{k}(m) > f_max
        f_max = f3Set{k}(m);
    end
end
f3Max(m) = f_max;

f_max = 0;
for k = 1:nRound
    if f4Set{k}(m) > f_max
        f_max = f4Set{k}(m);
    end
end
f4Max(m) = f_max;

f_max = 0;
for k = 1:nRound
    if fSet{k}(m) > f_max
        f_max = fSet{k}(m);
    end
end
fMax(m) = f_max;
end

% plot the results
figure(1);
subplot(2,2,1);
hold on;
plot(f1Mean,'color','m','LineWidth',2);
plot(f1Min,'color','b','LineWidth',1);
plot(f1Max,'color','r','LineWidth',1);
subplot(2,2,2);
hold on;
plot(f2Mean,'color','m','LineWidth',2);
plot(f2Min,'color','b','LineWidth',1);
plot(f2Max,'color','r','LineWidth',1);
subplot(2,2,3);
hold on;
plot(f3Mean,'color','m','LineWidth',2);
plot(f3Min,'color','b','LineWidth',1);
plot(f3Max,'color','r','LineWidth',1);
subplot(2,2,4);
hold on;
plot(f4Mean, 'color', 'm', 'LineWidth', 2);
plot(f4Min, 'color', 'b', 'LineWidth', 1);
plot(f4Max, 'color', 'r', 'LineWidth', 1);

figure(2)
hold on;
plot(fMean, 'color', 'm', 'LineWidth', 2);
plot(fMin, 'color', 'b', 'LineWidth', 1);
plot(fMax, 'color', 'r', 'LineWidth', 1);