Satellite range scheduling with the priority constraint: An improved genetic algorithm using a station ID encoding method

Li Yuqing a,*, Wang Rixin a, Liu Yu b, Xu Minqiang a

a Deep Space Exploration Research Center, Harbin Institute of Technology, Harbin 150080, China
b Harbin University of Commerce, Harbin 150028, China

Received 2 June 2014; revised 10 September 2014; accepted 5 March 2015
Available online 16 April 2015

Keywords
Genetic algorithm;
Ground space scheduling;
Priority;
Satellite range scheduling;
Space communication

Abstract Satellite range scheduling with the priority constraint is one of the most important problems in the field of satellite operation. This paper proposes a station coding based genetic algorithm to solve this problem, which adopts a new chromosome encoding method that arranges tasks according to the ground station ID. The new encoding method contributes to reducing the complexity in conflict checking and resolving, and helps to improve the ability to find optimal resolutions. Three different selection operators are designed to match the new encoding strategy, namely random selection, greedy selection, and roulette selection. To demonstrate the benefits of the improved genetic algorithm, a basic genetic algorithm is designed in which two cross operators are presented, a single-point crossover and a multi-point crossover. For the purpose of algorithm test and analysis, a problem-generating program is designed, which can simulate problems by modeling features encountered in real-world problems. Based on the problem generator, computational results and analysis are made and illustrated for the scheduling of multiple ground stations.

© 2015 The Authors. Production and hosting by Elsevier Ltd. on behalf of CSAA & BUAA. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Satellites support many important services, such as surveillance, geodesy and navigation, remote sensing and monitoring, telecommunications, data relay, and so on. All these services need frequent communications between satellites and ground stations. These communications include but are not limited to satellite orbit measurement, maneuver, commands upload, data download, and health and maintenance operations. The scheduling of these communication requests is called satellite range scheduling problem. With the increase of space exploring activities, more and more spacecraft have been launched to the Earth orbit, but the number of ground stations is rather limited. The result is that the scheduling of daily communications between spacecraft and ground stations is becoming a challenging problem.

To solve the contradiction between the limited ground stations and the increasing requirements of satellite measurement...
and control, researchers have studied the problem from different perspectives.

Gu et al. studied a tracking telemetry and command (TT&C) resources scheduling technique based on inter-satellite link (ISL). The TT&C data of ground facilities could be transmitted through ISL, so that the task burdens of ground TT&C devices could be reduced. However, this technique is limited to that all the satellites must be in the same network, for example, a constellation.

The arrangement about communication requests of satellites not in constellations still rely on scheduling and optimizing the communications between satellites and ground stations.

To pursue this goal, various planning and scheduling algorithms have been presented to give assistance to human schedulers, or maybe take place of them. Some examples are combinatorial optimization, constraint-based programming, heuristic algorithms, and genetic algorithms (GAs). However, there are several limitations in the existing works. Firstly, most of these researches are focused on how to maximize the number of served communication requirements between spacecraft and ground stations, but not considering the various priority levels which different tasks have. In reality, different communication requirements may have different priority levels. For example, the communication for health and maintenance should have a higher priority, as it may affect the safety of a spacecraft, while the communication for payload data download may have a lower priority, because this kind of task is more flexible in operation. If only aiming to maximize the task number, the tasks with a higher priority may be lost. In extreme conditions, this may lead to some serious damages. Secondly, the encoding method in the mathematical model is not well studied. Most of the studies adopt the encoding method that arranges the communicating tasks one by one in a timely order. This encoding method does not make use of the special information of the satellite scheduling domain. Thirdly, due to the limited encoding method above, the selection operator in a GA is bound to fail in making full use of the useful information in the satellite range scheduling problem. This will lead to a low efficiency of optimization.

To tackle these problems, this paper takes task priority into account and presents an adapted GA which adopts a new chromosome encoding method, based on arranging elements by the ground station ID order. With this innovative encoding method, three new selection operators are proposed (random selection, roulette selection, and greedy selection). In addition, this paper tests the new algorithm with different population sizes, different selection schemes, and different instance sizes. Meanwhile, in order to analyze the problem as realistically as possible, this paper focuses on a specific sub-problem called multi-resource range scheduling problem (MuRRSP), which is NP-hard.

The paper is organized as follows. The definitions and formulation of the general satellite range scheduling problem with the priority constraint are introduced in Section 2. In Section 3, considering the priority constraint, this paper modifies the basic GA for the problem. An evolutionary algorithm called station coding based genetic algorithm (SCBGA) is designed in Section 4, derived from some conceptions of the genetic algorithm and based on a new chromosome encoding method. Results are reported and discussed in Section 5. Finally, general conclusions and research avenues are drawn in Section 6.

2. Terminology and mathematical statement

As mentioned in Ref. scheduling daily communications between satellites and ground control stations is getting very difficult, since an increasing number of satellites must be controlled by a small set of ground stations. Consequently, the problem is oversubscribed, and a large number of communication requests are unserved. Scheduling these requests in the MuRRSP is an NP-hard problem.

On the other hand, the MuRRSP has some unique characteristics as described below:

(1) A ground station could communicate with a satellite only when the satellite lies within the transmitting horizon of the ground station (also called time window or visibility window). In general, this happens periodically within a planning horizon. That means there are several communication chances between a ground station and a satellite.

(2) A satellite could communicate with any ground station if there is a visibility window between them. That means if a satellite fails to establish a communication link with a ground station, it could try to link with other available ground stations.

(3) For the characteristics above and depending on the mission and orbit, the number of visibility windows for each satellite is perhaps different. In other words, some satellites may have fewer communication opportunities than others in a planning horizon. In most existing researches, for the purpose of maximizing the number of served communication requirements, a satellite with fewer communication opportunities will be scheduled in priority. However, if the task priority constraint is taken into consideration, this method may no longer apply.

These characteristics provide additional flexibility to generate a communication schedule but increase solving difficulty at the same time. Especially when taking the priority constraint into account, the existing methods need to be modified to adapt the new situation. This section gives the necessary definitions and formulations to the MuRRSP with the priority constraint.

2.1. Tasks

In most existing studies, satellite scheduling tasks refer to the operations that require ground-to-space communications, such as observations, communications, maneuvers, imaging, taking measurements, uplinks, and downlinks. This definition is suitable for the problem maximizing the number of ground-space communications, but not for the problem maximizing the priority.

This paper defines each visibility window as a task, for that the priorities of different visibility windows may be different, even for the same satellite. This is different from most of the existing researches.
2.2. Priority of tasks

Generally, different satellites have different priorities. For example, military satellites, satellites in fault, and satellites serving for emergencies may have a higher priority than civil or normal ones.

Moreover, different visibility windows for the same satellite may be different in real space missions. For example, a visibility window may have a higher priority than others because the ground station which it is linked to has a suitable antenna angle.

In a real-world situation, the priorities of tasks would be given by a human scheduler as input information before starting the scheduling process.

2.3. Mission requirements

The number of communications is a main requirement of mission operation for each satellite. Depending on the mission and the situation of the satellite, the number required for links can possibly range from 0 to more than dozens per day. For study convenience, this paper uses 4 times a day as the number of linking requirements in Section 5.

Besides the communication number of times required, there are also other kinds of mission operation requirements, which can be found in Ref. 2.

2.4. Mathematical statement

Consider a set of satellites and a set of ground stations. Civilian and military customers submit their requests for communication with satellites. All the requests are expected to be serviced in a specific time period and satisfied with several constraints, e.g., a ground station can communicate with a satellite only when the satellite lies within the visibility window of the ground station.

In general, the MuRRSP with priority could be defined as follows:

\[ \text{MuRRS} = \{ S, G, T, P, W \} \]

where \( S \) is the set of satellites, \( G \) the set of ground stations, \( T \) the set of communication tasks, \( P \) the set of priorities, and \( W \) the set of visibility windows.

For describing the problem clearly, the following parameters are defined as follows:

The input are

- \( s\_\text{num} \), the number of satellites;
- \( g\_\text{num} \), the number of ground stations;
- \( t\_\text{num} \), the number of communication tasks;
- \( w\_\text{num} \), the number of total visibility windows;
- \( \text{sat}_i \), \( i \in [1, s\_\text{num}] \), the \( i \)th satellite in set \( S \);
- \( \text{ground}_k \), \( k \in [1, g\_\text{num}] \), the \( k \)th ground station in set \( G \);
- \( \text{task}_n \), \( n \in [1, t\_\text{num}] \), the \( n \)th communication task in set \( T \);
- \( \text{win}_m \), \( m \in [1, w\_\text{num}] \), the \( m \)th visibility window in set \( W \);
- \( \text{J}_{\text{sat}_i, \text{ground}_k} \), the visibility window quantity of \( \text{sat}_i \) on ground station \( \text{ground}_k \), which subjects to \( \sum_{k=1}^{g\_\text{num}} \sum_{i=1}^{s\_\text{num}} \text{J}_{\text{sat}_i, \text{ground}_k} \);
- \( p_{\text{sat}_j, \text{ground}_k} \), the priority of task \( \text{task}_{\text{sat}_j, \text{ground}_k} \);
- \( w_{\text{sat}_i, \text{ground}_k} \), the visibility window of task \( \text{task}_{\text{sat}_i, \text{ground}_k} \), which subjects to \( \text{w}_{\text{sat}_i, \text{ground}_k} = [\text{start}_{\text{sat}_i, \text{ground}_k}, \text{end}_{\text{sat}_i, \text{ground}_k}] \);

The output are

- \( \text{task}_{\text{sat}_i, \text{ground}_k} \), the \( i \)th task (visibility window) of \( \text{sat}_i \) on ground station \( \text{ground}_k \). \( \text{task}_{\text{sat}_i, \text{ground}_k} = 1 \), if the task is selected; otherwise, \( \text{task}_{\text{sat}_i, \text{ground}_k} = 0 \). This variable is the main output of the problem.

The assumptions are

In addition, in order to develop analyses and research easily, according to the practical circumstances, this paper makes the following assumptions:

1. The communication duration of low Earth orbit (LEO) satellites is equal to the length of the visibility window.
2. All the ground stations possess the same equipment and ability for accomplishing the communication tasks.
3. Some special cases are excluded.
4. The shift time of communication equipment is \( C_1 \), which is a constant.
5. The minimum communication duration is \( C_2 \), which is a constant.

Based on the above definitions and formulations, the mathematical statement of the MuRRSP with priority could be established as follows:

The objective function is

\[
\max \sum_{i \in [1, s\_\text{num}]} \sum_{j \in [1, J_{\text{sat}_i, \text{ground}_k}]} \sum_{k \in [1, g\_\text{num}]} \text{task}_{\text{sat}_i, \text{ground}_k} \cdot p_{\text{sat}_j, \text{ground}_k}
\]

As described in Eq. (2), the priorities of tasks are taken into account. This makes the existing solution algorithms or methods not suitable to the new problem, for the reason that the searching direction is changed from maximizing the number of tasks to maximizing the total priority of the served tasks.

It could confirm that the higher priority a task has, the greater probability it could be served.

Eq. (2) is subjected to the constraints as follows:

\[
\forall w_{\text{sat}_i, \text{ground}_k} \in W, \quad \text{if} \text{task}_{\text{sat}_i, \text{ground}_k} = 1, \quad \text{then} \text{end}_{\text{sat}_i, \text{ground}_k} - \text{start}_{\text{sat}_i, \text{ground}_k} \geq C_2
\]

\[
\forall w_{\text{sat}_i, \text{ground}_k}, w_{\text{sat}_i, \text{ground}_k} \in W, \quad \text{if} \quad \text{task}_{\text{sat}_i, \text{ground}_k} = \text{task}_{\text{sat}_i, \text{ground}_k} = 1 \quad \text{and} \quad \text{ground}_1 = \text{ground}_2, \quad \text{then} \quad w_{\text{sat}_i, \text{ground}_1} \cap w_{\text{sat}_i, \text{ground}_2} = \emptyset \quad \text{and} \quad \text{end}_{\text{sat}_i, \text{ground}_1} + C_1 < \text{start}_{\text{sat}_i, \text{ground}_2} \text{ or } \text{end}_{\text{sat}_i, \text{ground}_1} + C_1 < \text{start}_{\text{sat}_i, \text{ground}_2}
\]

\[
\forall \text{sat}_i, \in S, \quad \text{if} \quad \text{task}_{\text{sat}_i, \text{ground}_1} = \text{task}_{\text{sat}_i, \text{ground}_2} = 1, \quad \text{then} \quad w_{\text{sat}_i, \text{ground}_1} \cap w_{\text{sat}_i, \text{ground}_2} = \emptyset
\]
Eq. (3) shows that the visibility window must satisfy the requirement of the minimal communication duration. Eq. (4) indicates the fact that one ground station could not support two or more communication tasks simultaneously, and the spare time between two tasks must be enough for one equipment shift. Eq. (5) shows that one satellite could establish only one communication link at one time.

3. Genetic algorithm for MuRRSP

Several algorithms can be used to search for near-optimal solutions to optimization problems, including heuristic approach and local search, the Branch and Bound algorithm, time indexed formulation, and so on. Recent studies find that genetic algorithms are among most successful algorithms for solving hard problems.

Because of the outstanding advantages a GA has, researchers have applied it into satellites range scheduling problems, and obtained a series of relatively good results. The challenge here is adapting it to be suitable for tackling the priority constraint. This paper uses the following methods to match this demand.

3.1. Encoding

Aiming at the MuRRSP with priority constraint, this paper adopts the binary chromosome coding method. Let task\(_{j,\text{sat}_i}\) be the element in the chromosome. task\(_{j,\text{sat}_i} = 1\) means that the task is selected, and task\(_{j,\text{sat}_i} = 0\) means that the task is canceled. According to the definitions in Section 2, every visibility window is a task. Therefore, the length of the chromosome is

\[
\text{length}_{\text{chromosome}} = w \cdot \text{num}
\]  

(6)

Let chrom\(_{k,a}\) be the \(a\)th chromosome in the \(k\)th generation. Arranging the tasks in the satellite ID order, the chromosome could be described as follows:

\[
\text{chrom}_{k,a} = \{\text{task}^1_{\text{sat}_1}, \text{task}^2_{\text{sat}_1}, \ldots, \text{task}^h_{\text{sat}_1}, \text{task}^1_{\text{sat}_2}, \text{task}^2_{\text{sat}_2}, \ldots, \text{task}^1_{\text{sat}_k}, \text{task}^2_{\text{sat}_k}, \ldots, \text{task}^1_{\text{sat}_p}, \text{task}^2_{\text{sat}_p}, \ldots\}
\]  

(7)

where task\(_{j,\text{sat}_i}\) denotes the \(j\)th task of the \(i\)th satellite. The \(i\)th satellite has \(f_i\) tasks.

3.2. Objective function and fitness calculate

The purpose of this research is to insure the executions of tasks with high priorities. To pursue this goal, the objective function could be defined as Eq. (2) and the fitness could be defined as follows:

\[
F(\text{chrom}_{k,a}) = \sum_{i=1}^{w \cdot \text{num}} \sum_{j=1}^{f_i} \text{task}^j_{\text{sat}_i,\text{ground}_k} \cdot p_{\text{sat}_i,\text{ground}_k} ^j
\]  

(8)

3.3. Constraints and conflicts checking

A significant difference between the MuRRSP with priority and other optimization problems lies in that when the algorithm gets a chromosome permutation, in the initial population or some offspring, it may not be a feasible solution, for there are several constraints which may lead to a lot of conflicts. To be suitable for this characteristic, this paper modifies the GA to check conflicts after generating a chromosome permutation and before calculating the chromosome fitness. Section 2 defined the main constraints in MuRRS problems. The conflict resolution methods are listed as follows:

1. If 2 tasks require the same ground station simultaneously, cancel the lower priority one.
2. If 2 tasks require the same ground station simultaneously and the priorities are same, cancel the one which has more communicating chances.
3. If 2 tasks require the same ground station simultaneously and both the priorities and the communicating chances are the same, cancel one randomly.

3.4. GA scheme for MuRRSP

Based on the above modifications, the basic GA can be used to resolve the MuRRSP. The pseudocode is shown in Fig. 1.

The basic GA uses a single point crossover to generate offspring. This paper adapts it to 13 random crossover points to compare with the SCBGA, which will be discussed in Sections 4 and 5.

4. A station coding based genetic algorithm for MuRRSP

To improve the performance of the basic GA, this section presents an evolution algorithm called station coding based genetic algorithm (SCBGA) derived from some conceptions of the genetic algorithm. To tackle the priority constraint and get a better performance, this paper uses the following technologies.

4.1. Encoding in ground station ID order

Similarly, let task\(_{\text{sat}_i,\text{ground}_k}\) be the element in the chromosome, and the length of the chromosome is

\[
\text{length}_{\text{chromosome}} = w \cdot \text{num}
\]  

(9)

| 1. Begin |
| 2. Input the parameters |
| 3. Generate the initial population |
| 4. Check conflicts |
| 5. while (not meet the ending conditions) |
| 6. Roulette selection |
| 7. Crossover |
| 8. Mutation |
| 9. Check conflicts |
| 10. End |

Fig. 1 Pseudocode of basic GA for MuRRSP.
Moreover, to enhance the searching performance of the algorithm, this paper presents an encoding method based on arranging the elements by the ground stations, which is described in detail as follows:

**Step 1.** Arrange the ground stations in their ID order.
**Step 2.** For each ground station, find all the tasks it could support, and put them together.
**Step 3.** Form the chromosome permutation and finish the encoding.

After the three steps above, letting $\text{chrom}_{k,n}$ be the $n$th chromosome in the $k$th generation, the chromosome could be described as follows:

$$\text{chrom}_{k,n} = \{ \tau^k, n_{\text{ground}_1}, \tau^k, n_{\text{ground}_2}, \ldots, \tau^k, n_{\text{ground}_m} \}$$ (10)

where $\tau^k, n_{\text{ground}_i}$ denotes all the tasks supported by ground station $i$ in the satellite ID order, which is described in Eq. (11).

The chromosome permutation could be obtained as shown in Fig. 2.

$$\tau^k, n_{\text{ground}_i} = \{ \text{task}^1_{\text{sat}_1, \text{ground}_i}, \text{task}^2_{\text{sat}_1, \text{ground}_i}, \ldots, \text{task}^J_{\text{sat}_1, \text{ground}_i} \}$$ (11)

4.2. Offspring generation

The selection of individuals and parts to be crossed is an important aspect in the SCBGA as it impacts on the diversity and the convergence of the algorithm. Random selection, roulette selection, and greedy selection schemes have been proposed in this paper for selection operators.

![Illustration of the chromosome](image-url)
This paper uses a multipoint cross operator to generate offspring. The number of ground stations and the number of parents are not limited. It means that $g_{\text{num}}$ parents at most will be selected and crossed.

The management of the unscheduled tasks is another important aspect. After selection, cross, and mutation, the child’s chromosome has been changed and may be very different from that of the parents. Therefore, it is possible to insert new tasks into the child from the set of unscheduled tasks.

Based on the above operators, the pseudocode of the offspring generation could be given as shown in Fig. 3.

It can be seen from the above steps that all the chromosomes may be engaged to generate the offspring (the number of actually selected chromosomes is equal to $g_{\text{num}}$, the number of ground stations), and only one child is produced in each generation, taking place of the worst parent. With the effects of evolution, the population will converge towards the optimal solution.

### 4.3. SCBGA for MuRRSP

Based on the above modifications, the SCBGA can be used to resolve the MuRRSP. The pseudocode is shown in Fig. 4.

As described in Eq. (11) and Fig. 2, all the tasks, i.e., visibility windows, are arranged in the station ID order. With proper genetic operators, this encoding method could support a more efficient search than the existing encoding methods. The reason is that with the new encoding method, the algorithm could adopt an efficient selection and cross operator:
(1) With the new encoding method, the algorithm could select excellent gene segments from parent chromosomes directly, and the segments have strongly practical background, i.e., a gene segment expresses all the tasks performed by a ground station, as illustrated in Fig. 2.

(2) Based on selecting excellent gene segments from parents, these gene segments represent the excellent parts of all the parents, so the algorithm could cross these excellent gene segments and generate an excellent offspring, as illustrated in Fig. 5. In this way, the offspring inherits excellent genes from the parents.

(3) With the new encoding method, the cost of computing time would be saved. This is because the algorithm will get the information of all the tasks served by every ground station, as soon as the establishing of the chromosome is finished. It happens only once at initial chromosome development and only takes a few of CPU time cost. Otherwise, the algorithm must compute each task that belongs to each ground station one by one and do it at every iteration, which will waste a lot of time.

Otherwise, if encoding tasks in a chronological order, like most existing methods, the gene segment selected would be confused for containing tasks that belong to many ground stations. It would lead to an inefficient cross and make the genetic algorithm like a random search. In fact, this disorder cross operator effectively becomes a mutation operator with a large mutation probability. It could be proved from the numerical examples described in Section 5.

5. Numerical examples and results

In this section, to demonstrate the validity of the proposed SCBGA, several numerical examples are tested and computational results are analyzed.

5.1. Instance generator

For the purpose of the algorithm test and analysis, according to Refs. 2,18, we have designed a problem generating program by Matlab 2010a, which could produce problems by modeling features encountered in the real-world problems. (The problem generator could be obtained by E-mail form the corresponding author.)

Assume that all the schedules task requests are on a per-day basis. Therefore, the lower and upper bounds of the task request (visibility window) are restricted in the interval [1,1440], the number of minutes in a 24-hour period.

In the real-world satellite scheduling problem, there are mainly 4 types of requests, such as downloading data from satellites, transmitting information or commands between ground stations and satellites, maneuver, and checking the health and status of a satellite. These 4 types of tasks could be simulated by the instance generator and described as shown in Table 1.

To generate different size instances, users could specify the numbers of satellites, ground stations, and communication times at the beginning. The generator will produce suitable instance according to a user’s request and features or characteristics introduced above.

5.2. Results and analysis

5.2.1. Comparison between basic GA and SCBGA

Firstly, the performances of the basic GA and the SCBGA are compared. As mentioned in Section 3, to compare with the...
SCBGA, this paper adapts the basic GA from a single point crossover (GAs) to a multi-point crossover (GAm). Here, the number of cross points is 13, which is equal and subjected to the number of ground stations in the instance used.

A total of 20 independent runs of the two algorithms are performed, under the parameter configurations listed in Tables 2–4, on a 2.5 GHz Pentium Dual-Core CPU and 2 GB RAM computer. Table 5 gives the results independently.

Fig. 6 and Fig. 7 give one typical evolution curve of the GA and the SCBGA, respectively.

Obviously, the SCBGA could generate a much better result than the GA, and needs a shorter CPU time in the meantime, even GAm with an improvement by the multi-point crossover. Increasing the population size could not significantly help to escape the local optimal. This is due to that the SCBGA adopts the new encoding method so that it reduces the
complexity of the conflict checking, and the SCBGA could select and cross the parents’ gene segments as a station unit (as illustrated in Figs. 3 and 4). This mechanism could make the offspring inherit excellent genes from the parents and help to find a better solution. This numerical example could prove the analysis given in Section 4.3.

Fig. 8 shows the final schedule developed by these three algorithms. Each rectangle filled with different colors in the chart denotes a served task, and different colors stand for different ground stations, with their IDs printed on the top of the rectangle. From Fig. 8, we can draw a conclusion that the SCBGA may find a better solution than those found by GAs and GAm. One reason is that it can schedule more tasks to be performed, so the total priority of the served tasks may be increased. This conclusion can be proved by Table 5, which gives the total priority of these three algorithms.

5.2.2. Different population sizes

Furthermore, we consider the parameter of population size in the SCBGA from 10 to 200 and give details as shown in Figs. 9–11 (the number of iterations is 5000).

A total of 20 independent runs of the SCBGA have been performed (under the parameter configurations listed in Table 6) and average results are reported.

Fig. 9 (a) shows that the graphical representation of the evolution of the maximal fitness obtained at each generation with population sizes varying from 10 to 200, and in Fig. 9 (b), the mean value of fitness is averaged by 20 runs. As can be seen in Fig. 9, a small population size (less than 50 for our numerical instance) leads to a premature convergence and hardly escapes from the local optimum for the reason of poor diversity. With the population size increasing, the population diversity is enhanced, and benefits in finding the optimal solution. On the other hand, when the population size passes a certain size, the results of total fitness do not increase obviously but the converging speed is reduced. This is due to that a large population requires more time to transmit good genes to offspring when only one individual is replaced at one time. Obviously, the larger the population size is, the longer the time needed for converging is.

Fig. 10 shows the maximal fitness obtained by each run, with population sizes varying from 10 to 200. With the population size increasing, besides the maximal fitness growth, the range of the maximal fitness volatility becomes tight.

We study the increase of the execution time to see the effect of the increase of the population size. As can be seen from Fig. 9 (b), the execution time does not increase significantly, for the reason that the CPU cost depends mostly on the offspring generating, which is almost the same despite of how many parents are involved.

Fig. 12 represents the evolution process of the child chromosome generating. As can be seen from the figure, limited by the small population size, the child concentrates quickly into the local optimal solution. This drawback will be changed by a larger population size. On the other side, the same as

Fig. 7  Evolution curve of SCBGA.

Fig. 8  Final schedule by GAs, GAm and SCBGA.
Fig. 9 shows, an excessive population amount is not significantly conducive to finding the optimal solution, but reduces the converging speed.

5.2.3. Different selection schemes

On the instance above, this paper compares the different selection schemes in the SCBGA, which are random selection, greedy selection, and roulette selection. The parameters are set as listed in Table 7. For each method, 20 runs have been performed and the results are given in Figs. 13–15 and Table 8.

Considering Figs. 13–15 and Table 8, the following comments could be made:

(1) Different selection strategies influence the population diversity and the convergence. The roulette selection gives the best solution in most cases.

(2) The greedy selection makes a fast convergence but fails to keep the diversity of the population. On the other hand, although the random and roulette selections have a slower convergence, they get a relatively better diversity, which is good at escaping from the local optimal and finding a better solution.

(3) A larger population helps the three selection strategies to improve the performance, and the CPU time does not increase significantly, for the reason that the CPU cost depends mostly on the offspring generating, which is

Table 6 Parameter configurations of instances.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>From 10 to 200</td>
</tr>
<tr>
<td>Iteration steps</td>
<td>5000</td>
</tr>
<tr>
<td>Number of satellites</td>
<td>30</td>
</tr>
<tr>
<td>Number of ground stations</td>
<td>13</td>
</tr>
<tr>
<td>Communications of every satellite per day</td>
<td>4</td>
</tr>
</tbody>
</table>

Fig. 9 shows, an excessive population amount is not significantly conducive to finding the optimal solution, but reduces the converging speed.
almost the same because despite how many individuals are involved in the population, the algorithm only makes $g\_num$ choices to select the parents.

4. There is no significant difference in the CPU time for the three selection methods. Because the CPU cost mainly depends on the generation of offspring, and different

Fig. 12  Evolution process of the child chromosome with different population sizes.
Selection strategies make an identical number of choices, \( g_{\text{num}} \), in the process of offspring generation. The CPU time is slightly affected by the selection strategies.

### 5.2.4. Different instance sizes

To evaluate the SCBGA performance in large-scale problems, several instances have been tested with the parameters listed in Table 9.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>50, 100, 150</td>
</tr>
<tr>
<td>Iteration steps</td>
<td>5000</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
</tbody>
</table>

#### Table 7 SCBGA parameters for different selection schemes.

Fig. 13 Evolution curves of SCBGA with a population size of 50.

Fig. 14 Evolution curves of SCBGA with a population size of 100.

Fig. 15 Evolution curves of SCBGA with a population size of 150.
Using the provided generator, three different instances with 30 satellites, 50 satellites, and 100 satellites are generated. For each instance, 20 runs have been performed and the results are shown in Figs. 16, 17 and Table 10.

As can be seen from Figs. 16, 17 and Table 10, the roulette selection gives the best solution in all the cases. In addition, from Figs. 16, 17, we could know that the roulette selection has a faster converging speed than the random selection method, which performance is better than the greedy strategy.

### Table 8 Results obtained with 3 selection methods.

<table>
<thead>
<tr>
<th>Population size</th>
<th>Max</th>
<th>Mean</th>
<th>Min</th>
<th>CPU mean time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>15126</td>
<td>14864.5</td>
<td>14681</td>
<td>568.9</td>
</tr>
<tr>
<td>100</td>
<td>15241</td>
<td>15010.8</td>
<td>14874</td>
<td>569.6</td>
</tr>
<tr>
<td>150</td>
<td>15357</td>
<td>15205.1</td>
<td>14992</td>
<td>545.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Population size</th>
<th>Random</th>
<th>Greedy</th>
<th>Roulette</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>15131</td>
<td>15176</td>
<td>14926.4</td>
</tr>
<tr>
<td>100</td>
<td>15173</td>
<td>15167.3</td>
<td>15348</td>
</tr>
<tr>
<td>150</td>
<td>15283</td>
<td>15249.5</td>
<td>15372</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Population size</th>
<th>Max</th>
<th>Mean</th>
<th>Min</th>
<th>CPU mean time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>15126</td>
<td>14864.5</td>
<td>14681</td>
<td>568.9</td>
</tr>
<tr>
<td>100</td>
<td>15241</td>
<td>15010.8</td>
<td>14874</td>
<td>569.6</td>
</tr>
<tr>
<td>150</td>
<td>15357</td>
<td>15205.1</td>
<td>14992</td>
<td>545.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Population size</th>
<th>Random</th>
<th>Greedy</th>
<th>Roulette</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>15131</td>
<td>15176</td>
<td>14926.4</td>
</tr>
<tr>
<td>100</td>
<td>15173</td>
<td>15167.3</td>
<td>15348</td>
</tr>
<tr>
<td>150</td>
<td>15283</td>
<td>15249.5</td>
<td>15372</td>
</tr>
</tbody>
</table>

### Table 9 SCBGA parameters for different instances.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>150</td>
</tr>
<tr>
<td>Iteration steps</td>
<td>5000</td>
</tr>
<tr>
<td>Selection operator</td>
<td>Random; Greedy; Roulette</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
<tr>
<td>Number of satellites</td>
<td>30, 50, 100</td>
</tr>
<tr>
<td>Number of ground stations</td>
<td>13</td>
</tr>
<tr>
<td>Communications of every satellite per day</td>
<td>4</td>
</tr>
</tbody>
</table>

Using the provided generator, three different instances with 30 satellites, 50 satellites, and 100 satellites are generated. For each instance, 20 runs have been performed and the results are shown in Figs. 16, 17 and Table 10.

As can be seen from Figs. 16, 17 and Table 10, the roulette selection gives the best solution in all the cases. In addition, from Figs. 16, 17, we could know that the roulette selection has a faster converging speed than the random selection method, which performance is better than the greedy strategy.

**Fig. 16** Evolution curves of SCBGA about the instance of 50 satellites.

**Fig. 17** Evolution curves of SCBGA about the instance of 100 satellites.
6. Conclusions

Scheduling the communications between satellites and ground stations is an important problem that arises in space mission operations, for the purpose of the efficient management of space missions. The problem is to efficiently allocate a large number of missions to a rather small number of ground stations. The problem is highly complex and over-constrained. This paper presents the implementation of a station coding based genetic algorithm for the MuRRSP, particularly with the priority constraint. To tackle with the priority and improve the performance, it adopts a new chromosome encoding method based on arranging the tasks in the ground station ID order. We have evaluated the SCBGA using the simulation case generated by the Matlab program. In our future work, we will fully evaluate the SCBGA and compare it to other meta-heuristics implementations for the problem.

References

Satellite range scheduling with the priority constraint: An improved genetic algorithm using a station ID encoding method


Li Yuqing received his B.S., M.S., and Ph.D. degrees from Harbin Institute of Technology in 2002, 2004, and 2008, respectively. Now he is an instructor there. His main research interests are planning and scheduling, satellite range scheduling, and autonomous spacecraft.

Wang Rixin is an associate professor at Harbin Institute of Technology. His area of research includes planning and scheduling, satellite range scheduling, and autonomous spacecraft.

Xu Minqiang is a professor and Ph.D. advisor at Harbin Institute of Technology. His current research interests are planning and scheduling, satellite range scheduling, and autonomous spacecraft.