2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Agile Earth Observing Satellites Mission Planning Using Genetic Algorithm Based on High Quality Initial Solutions

Zang Yuan*†, Yingwu Chen* and Renjie He*
*College of Information System and Management
National University of Defense Technology
Changsha, China 410073
†Email: zang.yuan@nudt.edu.cn

***Abstract*—This paper presents an improved genetic algorithm to solve the agile earth observing satellite mission planning problem. We study how to rapidly generate high quality initial solutions, and four generation strategies are proposed. The effect of the settings of operator parameters on the performance of the algorithm is analyzed. The experiment results show that the genetic algorithm based on high quality initial solutions generated by Hybrid Random Heuristic Strategy (HRHS) is more effective in solving the agile satellite mission planning problem, but in a certain time cost. We expect that our results will provide insights for the future application of genetic algorithm to satellites mission planning problems.**

I. Introduction

Earth Observing Satellites (EOS) play an important role in various fields of nowadays' society, such as mapping, environmental monitoring. The mission of EOS is to acquire photographs of specified areas on earth surface to satisfy the requirements of different users. The mission planning process of EOS mainly refers to the selection of a set of tasks and arrangement of their sequence to maximize the sum of revenue. Different from traditional satellites, Agile Earth Observing Satellites (AEOS), such as the American Ikonos satellite and the French Pleiades ones, have three degrees of freedom direction, i.e., roll, pitch and yaw, dramatically improving its observing ability. However, such ability is at the sacrifice of increasing the complexity of the mission planning process. For instance, the observing time windows become much longer, and the number of ways of observing a target on the Earth surface may be infinite, for starting times of the observation are free. As a result, agile satellite can observe much more tasks than non-agile satellite. As shown in Fig. 1, non-agile satellite can only complete three observing tasks while agile satellite can complete all the five observing tasks because of its long time window. It turns the EOS mission planning problem from only targets selection into both targets selection and observing time determination. This leads to significantly expanded feasible solution space and the more difficulty of searching solution[1]. Thus, traditional scheduling method for non-agile satellites is no longer applicable for agile satellites.

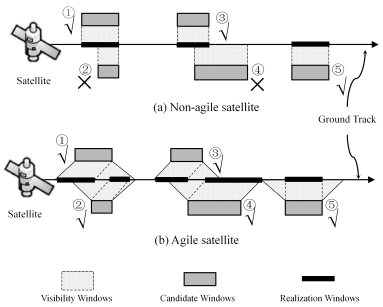

Fig. 1. Non-agile EOS and Agile EOS

Lots of researches studying the agile satellites mission planning problems have been reported in the literature. Wolfe et al.[2] introduced their models after summarizing the EOS scheduling problem. They proposed three methods to solve the EOS scheduling problem: a priority dispatch method, a look-ahead algorithm and a genetic algorithm. The priority dispatch method is fast and simple. They used it to produce acceptable schedules most of the time. The look-ahead algorithm redefines the best location rule to look ahead in the sequence of tasks which are not scheduled. The genetic algorithm can create near-optimal results but it takes more time. The representation of the genetic algorithm is based on permutation, so it is easier to execute mutation and crossover operations.

Verfaillie et al.[3] studied task acceptance and scheduling problem of AEOS for single satellite in PLEIADES project. They built mathematical model and analyzed its NP-hard feature. The paper presented different methods which had been investigated in order to solve a simplified version of the complete problem: a fast greedy algorithm, a dynamic programming algorithm, a constraint programming approach and a local search method. Their results showed that dynamic programming algorithm is better and fast, greedy algorithm is inferior but fast, constraint programming approach is also inferior and local search method is best when all constraints were taken into consideration.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

Mancel et al.[4] built integer programming model for PLEIADES in France and solved it by column-generation algorithm. The results showed that the performance is similar to local search of Verfaillie when the data is small, but bad when the data is big.

Bistra Dilkina et al.[5] approached a method combining search based on arrangement and constraint propagation to solve the AEOS problems. They used hill-climbing method and simulated annealing to generate new arrangement first, then determined if a task can be scheduled the arrangement by constraint propagation.

Djamal Habet et al.[6, 7] solved single agile satellite planning problems using tabu search algorithm. The tabu search algorithm explores a search space by consistent and saturated configurations. They used effective constraint propagation to ensure each new move is consistent in the search procedure. They introduced a secondary problem: minimization of the sum of the transition durations in a schedule in order to obtain better solution. They also gave upper bounds on a relaxed problem by dynamic programming algorithm.

Beaumet at al.[8] proposed their work about autonomous decision making on board for AEOS equipped with a cloud detection instrument. An online decision-making problem was described to manage on board and a reactive/deliberative architecture was provided. They also introduced an iterated stochastic greedy search algorithm in the deliberative part.

We notice that few papers deeply researched genetic algorithm on solving AEOS mission planning problems. Genetic algorithm(GA) is an optimization heuristic based on stochastic search, which is inspired by the biological process of natural selection[9, 10]. They have been used to solve a wide range of combinatorial optimization problems such as planning and scheduling, traveling salesman problem (TSP), vehicle routing problem (VRP), layout design and so on. GA has been used to effectively solve a large number of complex problems with multiple objectives and constraints[11–13].

In this paper, we use an improved genetic algorithm based on high quality initial solutions generated by different strategies to solve the agile earth observing satellite mission planning problem. The parameters are determined by experiments and compare it with heuristics and standard simulated annealing. The results show that this method can improve the performance of the algorithm for AEOS mission planning problems.

II. Model

A. Problem Description and Assumption

In this paper, we consider a single satellite in one track, during which each target can only be observed once. A satellite can only observe a target during the limit time, visibility window. However, there are too many target observing tasks to be complete for the satellite. Then, the planning process need to decide which tasks are selected and when to complete the observing tasks.

B. Mathematical Model

The AEOS mission planning problem considered in this study can be defined formally as follows. For one satellite, there is a set of N observing tasks to be planned. Each task $i(i = 1, 2, \ldots, N)$ is identified with the following attribute of non-negative real number value:

t_i: observing time, which is the necessary minimum time it takes to complete the task;

t_{esi}: earliest start visibility time, after which the target can be observed;

t_{lei}: latest end visibility time, after which the target cannot be observed;

t_{lsi}: latest start observing time, which can be defined by $t_{lsi} = t_{lei} - t_i$;

t_{vi}: visibility time, which is equal to $t_{lei} - t_{esi}$;

t_{si}: real start observing time, which must between t_{esi} and t_{lsi};

r_i: the revenue obtained from the completion of task i.

Given the above definitions, the objective is to find an observing sequence of all selected tasks for one satellite that maximizes the total revenue R, which can be formulated as:

$$maxR = \sum_{i=1}^{N} x_i r_i \tag{1}$$

where x_i is an indicator that equals to 1 if task i is selected, and 0 otherwise.

Despite there are many practical constraints for real application[14], in this research, we only consider the following three constraints.

1) Planning Period Constraint: This constraint ensure that all planned execution of tasks must be in the planning period. Specifically, the start time of task execution cannot be earlier than the planning period start time T_s and the finish time of the task execution cannot be later than the planning period end time T_e. The constraint is represented as follows:

$$T_s \leq t_{si} < t_{ei} \leq T_e \tag{2}$$

2) Task Time Window Constraint: This constraint indicates that any task to be executed must be in its time window. It means that the execution of task cannot be started before its earliest start time or after its latest start time. The constraint is represented as follows:

$$t_{esi} \leq t_{si} \leq t_{lei} \tag{3}$$

3) Duration constraint: This constraint represents that at any time point one satellite can only perform one task. In other words, no overlapping exists between two tasks. The constraint is represented as follows:

$$t_{si} + t_i \leq t_{sj} (i \neq j \text{ and } t_{si} \leq t_{sj}) \tag{4}$$

Thus, if the start time of task j is not earlier than the task i start time, the start time of task j is not earlier than the end time of task i.

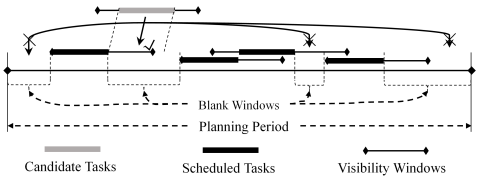

Fig. 2. Try to Schedule a Task

III. ALOGRITHM

The simple genetic algorithm (SGA) can be defined as a tuple consisting of eight elements:

$$SGA = (C, E, P_0, M, \Phi, \Gamma, \Psi, T) \tag{5}$$

where C is the coding mode for individual; E is the fitness value; P_0 is the initial solutions; M is the population size; Φ is the selection operator; Γ is the crossover operator; Ψ is the mutation operator and T is the terminal condition.

A. Coding Mode

As each task will not be observed repeatedly, our coding mode is learnt from the use of genetic algorithm for traveling salesman problem (TSP). For a problem with tasks, we use a string of positive integers sequence from 1 to N to represent an individual's chromosome, which signifies the order of tasks to be scheduled. Thus, different orders of the positive integers represent different individuals. For example, there are five tasks (from 1 to 5) to be planned. A string {1,2,3,4,5} represents a chromosome and a string {3,2,5,4,1} represents another chromosome.

B. Decoding Mode and Fitness Value

We use the total revenue R as an individual's fitness value. When we evaluate an individual's fitness value, we try to schedule all the tasks in accordance with the sequence of its chromosome. We make x_i equal to 1 if task i can be completed without constraints violations, otherwise x_i equal to 0, and then try to schedule the next task in the sequence until the last task.

As shown in Fig. 2, when we try to schedule a candidate task i (the gray one in the figure), we try to insert the task into each blank window. From the figure we can see that, task time window constraints are not met for 1st, 3rd and 4th blank window, while all constraints are met for 2nd blank window. Thus we make x_i equal to 1 and the real start time is set by the end time of the 1st scheduled task. Then we schedule next task in the string until all tasks have been tried to schedule and the end of the sequence is reached.

After decoding a chromosome, the fitness value (total revenue R) can be calculated according to Equ. (1). For example, there is a chromosome string $\{3, 2, 5, 4, 1\}$. We schedule the task 3 at first and then try to schedule task 2, task 5,..., until task 1. If string $\{3, 5, 4\}$ is decode, means that task 3, 5, and 4 can be executed in practice and the revenue $R = r_3 + r_5 + r_4$.

C. High Quality Initial Solutions

Like other intelligent optimization algorithms, GA begin with an initial set of solutions and find the optimal solution at last by iteration. So the quality of the initial population directly impacts the efficiency of the algorithm. Generally, for an algorithm, we expect the solving process is fast and the obtained solution is good. However speed and result are usually contradictory. Therefore, how to effectively generate high quality initial solutions is a problem worthy of further study for most intelligent optimization algorithms including GA.

In this paper, high quality initial solutions is defined as follows. This initial solutions in the population must have high average fitness value R and high population diversity, which can be described by variance σ^2 (or range d) of individual fitness value. On the other hand, we must take the generation time t into consideration, since the time is more important for satellite mission planning because of its emergency.

We propose four strategies to generate initial solutions:

1) Random Strategy(RS): Generate M strings of positive integers sequence randomly and use the strings as the initial solutions directly, which is like the SGA.

2) Random No Duplicates Strategy(RNDS): In this paper, we assume that the individuals with the same fitness value are duplicate. So we generate a string of positive integers sequence randomly and calculate the fitness value at first. The new generated string is added into the population if the fitness value is not equal to any other fitness value of existing initial solutions. Otherwise the string is discarded and the process above is repeated until the number of initial solutions in the set reach M.

3) Hybrid Random Heuristic Strategy(HRHS): In this strategy, half of the initial solutions are generate by random strategy and the others are generated by mixing heuristic. At first, half of the initial solutions are generate by random strategy. After that we generate other half of initial solutions by mixing original initial solutions as follows:

Step 1: Select a solution S_i randomly from the set of original initial solutions by selection probability which is according to their fitness value.

Step 2: Select the first task form S_i, put it into the string sequence of new solution S_{new} if the task is not in S_{new}, otherwise select the next task until a task is added into S_{new}.

Step 3: Repeat Step 1 and 2 until all the tasks have been added into S_{new}. Thus a new initial solution is generated.

Step 4: Repeat Step 1 to 3 until the number of initial solutions reach M.

For example, we generate three original initial solutions by three heuristic: S_1={1,2,3,4,5|R=3}, S_2={3,2,5,1,4|R=5} and S_3={4,2,1,5,3|R=2}. As shown in Fig. 2, $S_2$is selected first and task 3 is put into S_{new}. Then S_3 is selected second and task 4 is put into S_{new}. And so on, task 2,1 and 5 is put into S_{new} respectively.

Here we define six heuristic based on greed rules:

Greed Rule 1: Arrange the task in an ascending sort order of earliest start time.

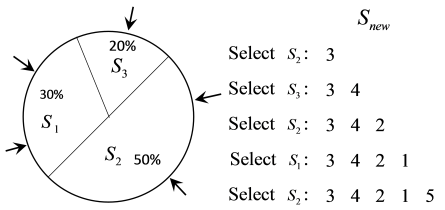

Fig. 3. An Example to Generate Initial Solutions by HRHS

Parent Individual 1	2 9 1 5 4 8 3 7 6
Parent Individual 2	8 9 6 3 4 1 2 7 5
Random Cross Location	* *
Copy the same gene in the same location	9 4 7
Copy gene in parent individual 1 between Cross Location	9 1 5 4 8 7
Copy the other gene in parent individual 2 in order	6 9 1 5 4 8 3 7 2

Fig. 4. An Example for Same-Sit-Copy-First Principle

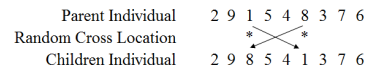

Fig. 5. Mutation Operator by Exchange of Two Gene Location

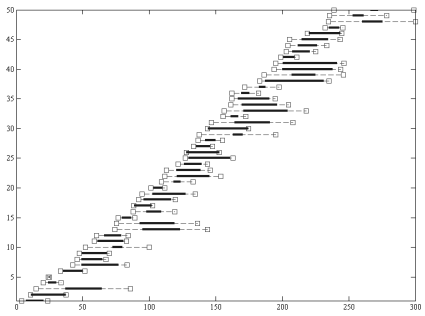

Fig. 6. Tasks in Test Problem with 50 tasks

Greed Rule 2: Arrange the task in a descending sort order of revenue.

Greed Rule 3: Arrange the task in an descending sort order of unit revenue $\tilde{r}_i = r_i/t_i$.

Greed Rule 4: Arrange the task in an ascending sort order of observing time.

Greed Rule 5: Arrange the task in an ascending sort order of visibility time.

Greed Rule 6: Arrange the task in an ascending sort order of blank visibility time $t_{bvi} = t_{vi} - t_i$.

4) Filtrate Strategy(FS): Generate $M^+ = wM$ (w is a positive integer) solutions by random strategy and sort them by their fitness value in descend order. Then select the M solutions every w solutions as initial solutions at the beginning with 1st solution.

D. Genetic Operators

1) Selection Operator: In selection operator, we select a pair of solutions from parent randomly and then preserve the solution with greater fitness value as a next generation individual until the number reaches the population size.

2) Crossover Operator: In this paper, we use the crossover operator based on Same-Sit-Copy-First Principle[15] by the crossover probability P_c, which can keep the gene in both absolute location and relative location.

For example in Fig. 4, we select two parent individuals according to P_c and two cross locations randomly. At first, we copy the gene in the same location if they are the same (such as 9,4 and 7). Then we copy the gene from parent individual 1 between random cross location (such as 1,5 and 8). At last, we copy the other gene in parent individual 2 as its sequence. Thus a new individual is generated.

3) Mutation Operator: In this paper, we select a solution by the mutation probability P_m and exchange the values of two randomly chosen gene locations.

For example in Fig. 5, we select a parent individual and two cross location randomly. Then we generate the new children individual by exchanging the gene in the cross location.

E. Termination Rule

The algorithm is terminated when the number of iterations reach the maximum number of iterations$G_{\max}$.

IV. Results and Discussion

A. Test Problems

In the experimental study, test problems were generated randomly. We generated 5 sets of test problems with the same planning period T and the number of task are 10, 30, 50, 70 and 90, respectively. Tasks in each set of test problems can be generated as follows. The earliest start visibility time t_{esi} is drawn from a uniform distribution$(0, T)$. The observing time t_i is drawn from a normal distribution (6,1). The latest start visibility time t_{lsi} equals to earliest start time adding the absolute value of the random number drawn from a normal distribution (5,1). The revenue p_i is drawn from a uniform distribution (0,1). Tasks in each set of test problems are numbered by earliest start time order.

Fig. 6 shows tasks distribution with $N = 50$. The vertical axis represents time (in seconds) and the horizontal axis represents the number of tasks. The solid lines represent t_i and the dashed lines mean the time windows. We can see the 50 tasks during the planning period and violent conflict among them from the figure intuitively.

B. Parameter Study and Results

In this section, we study the effect of different parameters (population size M, crossover probability P_c and mutation probability P_m) on the algorithm by controlling variables and find the best parameter combination ($M = 100$, $P_c = 0.8$ and $P_m = 0.0006$). Here we use the test problem with $N = 50$ and generate initial solutions by Random Strategy.

In order to get the reliable experiment results, we repeat the experiments ten times for each study. We compare the results in terms of average total revenue $\bar{R}$, the number of completed tasks $\bar{N}_c$ and processing time $\bar{T}_p$ (in CPU seconds). Test programs are coded in MATLAB 2010a and ran on a Lenovo computer with an Intel Core 2 1.66GHz processor.

TABLE I
RESULTS OF DIFFERENT M

M	$\bar{R}$	$\bar{N}_c$	$\bar{T}_p$
10	6.9644	9	34.288
50	8.4787	10.3	170.306
100	10.4822	14.7	349.098
150	11.0184	14.8	523.069
200	11.399	15	672.423

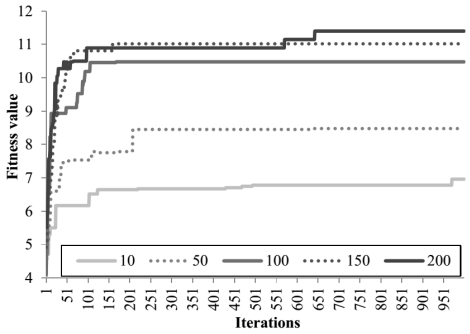

Fig. 7. The Evolution Curve in Different M

TABLE II
RESULTS OF DIFFERENT P_c

P_c	$\bar{R}$	$\bar{N}_c$	$\bar{T}_p$
0.0	8.2738	11.0	45.271
0.2	8.0114	11.3	77.313
0.4	8.8428	13.4	109.98
0.6	9.7004	13.8	146.203
0.8	99975	13.2	174.393
1.0	8.9885	12.4	201.256

1) Population Size: We keep other parameters the same and repeat the experiments with M= 10, 20, 50, 100, 150, 200. The results and evolution curve are shown in Table I and Fig. 7, respectively.

From the table and figure we can see that population size has a great influence on the results. As the population size increases, convergence speed and total revenue increase, while processing time becomes longer. Since the population size is positively related to processing time, there is a tradeoff between results and processing time when we are to determine the population size. The figure shows that the increase speed of total revenue decreases gradually as the population size increases. Therefore we set M=100 in terms of time and performance.

2) Crossover Probability: We keep other parameters and repeat the experiments with P_c = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0. The results and evolution curve is shown in Table II and Fig. 8.

We can see that crossover probability has a great influence on the results but they are not monotonous related. As the crossover probability increases, the total revenue first rises and then falls and processing time takes longer. So there is a tradeoff between results and processing time when we are to determine crossover probability. In this sample the result is best with P_c = 0.8.

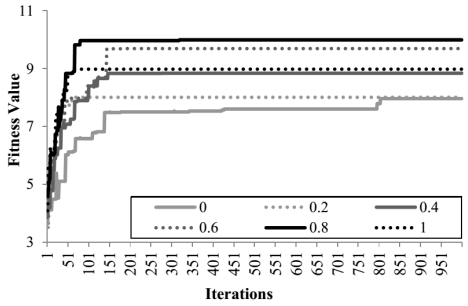

Fig. 8. The Evolution Curve in Different P_c

TABLE III
RESULTS OF DIFFERENT P_M

P_m	$\bar{R}$	$\bar{N}_c$	$\bar{T}_p$
0.000	5.377	8.4	120.246
0.002	8.7866	12.2	138.435
0.004	9.282	12.3	153.27
0.006	9.8138	13.1	163.051
0.008	7.7393	10.4	176.343
0.010	6.7996	9.3	201.693

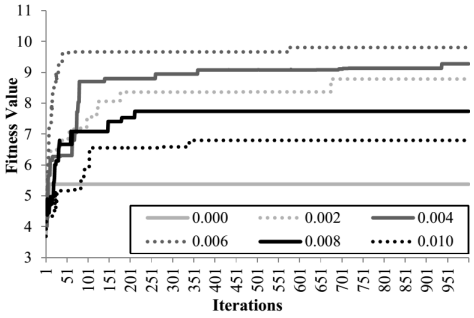

Fig. 9. The Evolution Curve in Different P_m

3) Mutation Probability: We keep other parameters the same and repeat the experiment with P_m = 0.000, 0.002, 0.004, 0.006, 0.008, 0.010. The results and evolution curve is shown in Table III and Fig. 9.

We can see that mutation probability has a huge influence on the results but mutation probability and fitness value are not monotonous related, too. As the mutation probability increases, the total revenue first rises and then falls and processing time becomes longer. When mutation operator is not used (P_m = 0.000), the algorithm loses the ability of search. So there is also a tradeoff between results and processing time when we are to determine the mutation probability. In this sample the result is best with P_m = 0.006.

C. Solutions Initialization Strategy Study and Results

In this section, we run the algorithm with four different strategies to generate initial solutions in order to study whether

TABLE IV
RESULTS OF DIFFERENT SOLUTIONS INITIALIZATION STRATEGIES

Strategy	$\bar{T}_g$	Fitness Value				
		Mean	Max.	Min.	Range	Var.
RS	0.204	8.0417	11.4002	4.3578	7.0425	1.9448
RNDS	0.213	8.0297	11.2550	4.4622	6.7928	1.7586
HRHS	0.319	9.6745	13.1538	4.6106	8.5432	4.8844
FS[a]	3.827	8.1717	12.7482	4.8444	7.9038	1.8541
FS[b]	37.513	8.1892	13.1835	4.9801	8.2034	1.9370

[a] means select 100 solutions from 1000 solutions.
[b] means select 100 solutions from 10000 solutions.

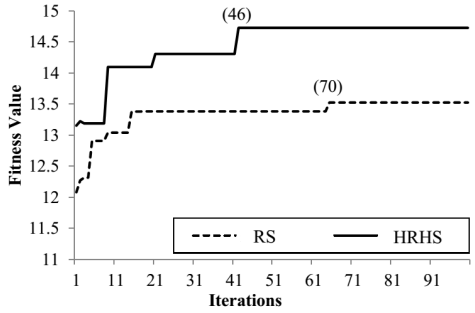

Fig. 10. The Evolution Curve in RS and HRHS

and how different solutions initialization strategies have an effect on algorithm performance. In order to get the reliable experiment results, we repeat the experiments ten times for each strategy. The results are shown in Table IV. In the table, $\bar{T}_g$ is the time of initial solutions generating. Mean, maximum and minimum values, range and variance of the fitness value are listed as well. Mean, maximum and minimum values can represent the quality of the initial solutions and range and variance can indicate their diversity.

From Table IV, we can see that, in HRHS, the average fitness value is the biggest, the population diversity is better (variance and range is bigger) and processing time is much shorter than FS. FS is better than first two strategies in average fitness value and diversity but it takes too much time. RS is slightly better than RNDS.

We draw the evolution curves of GA with HRHS and RS in Fig. 10. We can find that, for the algorithm with HRHS, the convergence speed is faster and result is better that with RS. So the GA with this strategy of generating initial solutions can find better solutions rapidly by smaller number of iterations.

D. Comparison Study and Results

In this section, the results obtained by genetic algorithm based on five strategies of initial solutions and simulated annealing (SA) are compared. The parameters of GA are $M = 100$, $P_c = 0.8$, $P_m = 0.006$, $G_{\max} = 50$, the test problems have 10, 30, 50, 70 and 90 tasks, respectively. We run ten times for each test set to ensure the reliable results. The results are shown in Table V.

From the table we can see that, for the small-scale data, all the algorithms and strategies have the same mean total revenue and number of completed tasks. However, for large-scale data, the algorithm with HRHS has greater advantages on total revenue and processing time.

TABLE V
RESULTS OF DIFFERENT ALGORITHM

Test Set	Algorithm	$\bar{R}$	$\bar{N}_c$	$\bar{T}_p$
10	RS	3.9782	10.0	9.923
	RNDS[a]	—	—	—
	HRHS	3.9782	10.0	9.746
	FS1	3.9782	10.0	10.908
	FS2	3.9782	10.0	20.722
	SA	3.9782	10.0	30.738
30	RS	8.1286	15.9	18.478
	RNDS	8.3283	16.1	19.266
	HRHS	8.4465	16.3	18.400
	FS1	8.3346	16.1	20.868
	FS2	8.3283	16.2	41.670
	SA	8.3549	16.2	51.330
50	RS	14.2060	22.3	27.553
	RNDS	14.3405	21.9	27.022
	HRHS	15.0225	23.8	27.960
	FS1	14.4068	22.7	30.760
	FS2	15.0070	23.1	63.954
	SA	14.8016	22.9	70.496
70	RS	20.9655	32.6	36.753
	RNDS	20.1638	32.4	36.893
	HRHS	21.1788	32.3	37.024
	FS1	21.1176	32.8	42.319
	FS2	20.3420	34.4	90.685
	SA	20.4024	30.7	95.117
90	RS	20.0290	33.4	44.783
	RNDS	20.0290	32.6	44.154
	HRHS	21.8689	37.6	46.260
	FS1	20.1245	33.4	49.726
	FS2	20.3433	33.9	109.420
	SA	20.7596	36.8	116.392

[a] For small-scale data, generating initial solutions with RNDS takes too long time so we discard the results.

V. CONCLUSION

In this paper, we present a genetic algorithm based on high quality initial solutions to solve the AEOS mission planning problem. Four strategies are designed to generate high quality initial solutions. Chromosome representation and genetic operators are designed for the problem. The sensitivities of parameters, such as population size, crossover and mutation rates, are investigated in the experiments. Experimental results show that the genetic algorithm with initial solutions generated by HRHS performs is better than others for AEOS mission planning problem. Experimental results also suggest that high quality initial solutions have a great effect on improving the performance of GA.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China (No.71101150 and 71101013).

REFERENCES

[1] G. Beaumet, G. Verfaillie, and M.-C. Charmeau, *Estimation of the minimal duration of an attitude change for an autonomous agile earth-observing satellite*. Springer, 2007, pp. 3–17.

[2] W. J. Wolfe and S. E. Sorensen, "Three scheduling algorithms applied to the earth observing systems domain," *Management Science*, vol. 46, no. 1, pp. 148–166, 2000.

[3] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology*, vol. 6, no. 5, pp. 367–381, 2002.

[4] C. Mancel and P. Lopez, "Complex optimization problems in space systems," in *13Th International Conference on Automated Planning & Scheduling (ICAPS'03)*, Conference Proceedings.

[5] B. Dilkina and B. Havens, "Agile satellite scheduling via permutation search with constraint propagation," 2005.

[6] D. Habet, M. Vasquez, and Y. Vimont, "Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite," *Computational Optimization and Applications*, vol. 47, no. 2, pp. 307–333, 2008. [Online]. Available: ¡Go to ISI¿://WOS:000281698100006

[7] D. Habet, *Tabu Search to Solve Real-Life Combinatorial Optimization Problems: A Case of Study*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, vol. 203, ch. 6, pp. 129–151.

[8] G. Beaumet, G. Verfaillie, and M.-C. Charmeau, "Feasibility of autonomous decision making on board an agile earth-observing satellite," *Computational Intelligence*, vol. 27, no. 1, pp. 123–139, 2011. [Online]. Available: ¡Go to ISI¿://WOS:000287401200008

[9] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[10] L. D. Chambers, *Practical handbook of genetic algorithms: complex coding systems*. CRC press, 2010, vol. 3.

[11] L. T. Bui, M. Barlow, and H. A. Abbass, "A multi-objective risk-based framework for mission capability planning," *New Mathematics and Natural Computation*, vol. 5, no. 02, pp. 459–485, 2009.

[12] J. Xiong, K. Shafi, and H. A. Abbass, "Multi-uncertainty problems (mup) with applications to managing risk in resource-constrained project scheduling," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, Conference Proceedings, pp. 1–8.

[13] J. Xiong, J. Liu, Y. Chen, and H. Abbass, "A knowledge-based evolutionary multi-objective approach for stochastic extended resource investment project scheduling problems," pp. 1–1, 2013.

[14] J. Liu, W. Zhong, and L. Jiao, "A multiagent evolutionary algorithm for constraint satisfaction problems," *IEEE Trans Syst Man Cybern B Cybern*, vol. 36, no. 1, pp. 54–73, 2006, liu, Jing Zhong, Weicai Jiao, Licheng eng Evaluation Studies Research Support, Non-U.S. Gov't 2006/02/14 09:00 IEEE Trans Syst Man Cybern B Cybern. 2006 Feb;36(1):54-73. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16468566

[15] L.-N. Xing, Y.-W. Chen, K.-W. Yang, F. Hou, X.-S. Shen, and H.-P. Cai, "A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1370–1380, 2008.