Optimization of Space Debris Collecting Mission

Eric G. Müller

Grupo VRI - Visão Robótica e Imagens
Universidade Federal do Paraná

June 4, 2018

Reference Paper

Multiple Space Debris Collecting Mission: Optimal Mission Planning[1]

- Publication:
  - Journal of Optimization Theory and Applications
  - October 2015, Volume 167, Issue 1, pp 195–218
- Authors:
  - Max Cerf
- Qualis:
  - A1 (2013-2016)

[1] M. Cerf. Multiple Space Debris Collecting Mission: Optimal Mission Planning. In Journal of Optimization Theory and Applications, 2015.

Problem Definition

Problem: The amount of space debris in orbit is increasing and must be collected to keep stable or even decrease their population. In order to collect these debris, a collection mission must be planned to reduce cost at maximum. Therefore, a mission optimization will create the best cost effective scenario to stabilize debris population (5 per year).

Figure: Image posted at SPACEREF[2] to show the space debris' problem.

[2] Space Debris is an Issue - Europe Wants Everyone to Deal with it

Detailing the Problem

At first glance, this problem seems a lot with the travelling salesman problem (TSP). But time is variable and constantly the position of the debris are changing. Therefore, the space debris collection problem (SDC) is different, as shown.

	TSP	SDC
Nodes visited	N	$m \times n \leq N$
Path definition	single closed path	multiple opened sub-paths
Node positions	fixed	moving
Edge valuation	fixed	time-dependent
Cost function	path-length	maximum sub-path cost

Table: TSP versus SDC differences.

Solution Method Proposed

Since this is a complex version of an already complex problem, the solution proposed was to use simulated annealing (SA) that has proven quite successful on large TSP instances.
The approach consists in using response surface modeling (RSM) based on cost matrices.

- The cost matrices store the costs of all the possible elementary transfers between debris for a mesh of discretzed dates.
- SA algorithm is applied using the possible transfer dates as costs.
- Trajectory optimization is them used on the debris order selected by SA solution in order to accommodate the mission in the best date and ensure the best maneuvers.

Restrictions

In order to keep the stability of debris population and reduce cost, 3 missions are planned using the same vehicle. Therefore, some constraints must be considered:

- All 3 mission must use the same vehicle, hence the maximum of those 3 sub-path cost used as solution and not the best one;
- Maximum mission duration allowed is 1 year;
- Number of debris to collect is 5.

Representation

Identifier	Description
N	Total number of debris
n	Number of debris visited per mission
m	Number of missions
K_i	Cost of the i^{th} mission
d_j	Position of the j^{th} debris
t_j	Date of visit of the j^{th} debris
t_0	Mission launch date

Table: List of identifiers.

Differences

Differences for this Proposed Work:

- Only 1 mission is computed;
- 5 debris are selected for collection;
- Maximum mission duration is restricted to 1 year (365 days), in order to keep the 5 collected debris per year rate;

Representation

A solution is defined by:

$S =$ Array with 5 positions containing the number of each debris

Each edge cost is determined by:

$C(t, j, j-1) =$ Transfer time from $(j-1)^{th}$ to j^{th} debris starting at time t

Fitness is determined by minimizing:

$$K = \sum_{j=0}^{4} C(t_j, j, j-1)$$

Cost Matrix Considerations

In order to calculate the time required for a transition bewtween debris position in each date a lot of effort and knowledge is required. Even using tools, such as Orekit[3], this task is rather complicated. To enforce how complex this calculations are, there are multiple papers aiming only on this issue.

Considerations:

- Launch Latitude = 0^o
- Launch Longitude = 0^o
- Launch Orbit = 0^o *Inclination*
- Launch Motion = 0 $Revs/day$
- Vehicle Acceleration = 0.1 $Revs/day^2$
- Vehicle Orbital Acceleration = $5^o/day^2$

[3] ORbits Extrapolation KIT https://www.orekit.org/static/index.html

TLE Elements

In order to simplify the calculations, only the following data is used:

Inclination	Orbit's latitudinal inclination (Equator)
Ascending Node	Orbit's longitudinal inclination (Greenwich)
Argument of Perigee	Position in degrees of the debris
Mean Motion	Mean speed of the debris

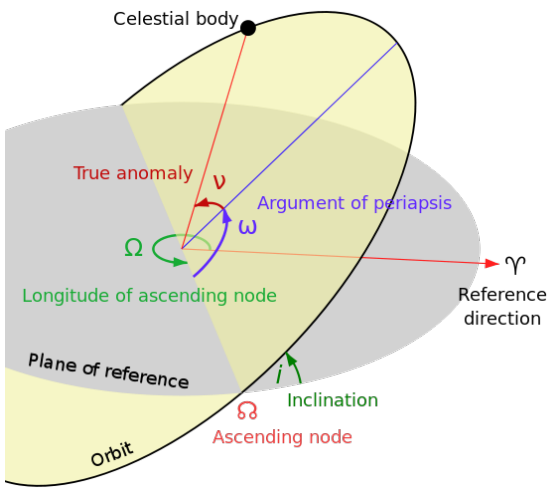

Figure: Image taken from Wikipedia[4] which demonstrates orbits information

[4] WIKIPEDIA - Longitude of the ascending node

Computing Transfer Time Cost

Step	Action / Condition
1:	$timeLatitude \Leftarrow Position2Time(Abs(\Delta Inclination))$
2:	$timeLongitude \Leftarrow Position2Time(Abs(\Delta AscendingNode))$
3:	$timeInOrbit \Leftarrow SpeedAndPostion2Time(\Delta Pedigee, \Delta MeanMotion)$
4:	$Time \Leftarrow Max(timeLatitude, timeLongitude, timeInOrbit)$

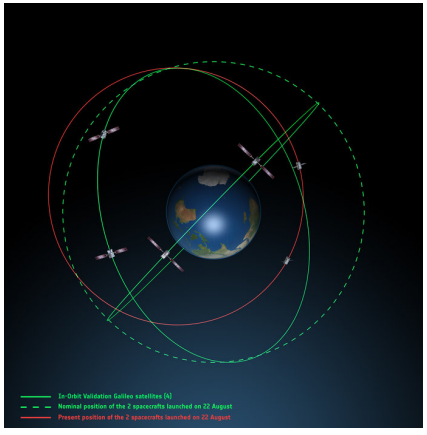

Figure: Image taken from GALILEO[5] as examples of orbits

[5] First Galileo FOC Satellite moving to new orbit

Algorithm 1 - Simulated Annealing

Step	Action / Condition
1:	$t \Leftarrow$ Initial Temperature
2:	$S \Leftarrow$ Random Initial Solution
3:	$Best \Leftarrow S$
4:	**repeat**
5:	**for** loop
6:	$R \Leftarrow$ Tweak(Copy(S))
7:	**if** $K(R) > K(S)$ or $rand(100\%) < e^{(K(R)-K(S))/t}$ **then**
8:	$S \Leftarrow R$
9:	Decrease t
10:	**if** $K(S) > K(Best)$ **then**
11:	$Best \Leftarrow S$
12:	**until** *Best* is ideal or time limit is reached
13:	**return** *Best*

Algorithm 1 - Simulated Annealing

Why use Simulated Annealing?

- Used in the reference paper. It will be compared as a close to the reference algorithm with other algorithm.
- Is said to be a good tool to solve complex travelling salesman problems.

Parameters:

- ξ_0 = Empiric value used to estimate initial temperature (0.8)
- α = Temperature Decay Rate
- *loop* = Iterations before decreasing temperature

Algorithm 1 - Simulated Annealing

Specific Processes:

- Initial Temperature = Proposed by Johnson et al., 1987, where 10 solutions are randomly generated and their fitness differences are used to estimate a good initial temperature

$$T_0 = -\Delta E^+ / ln(\xi_0)$$

- Tweak = Random change in a random index in the current solution to provide diversification

Algorithm 2 - Genetic Algorithm with Elitism

Step	Action / Condition
1:	$popsize \Leftarrow$ desired population size
2:	$n \Leftarrow$ desired number of elite individuals (even)
3:	$p \Leftarrow \{\}$
4:	**for** size of *popsize* times **do**
5:	$P \Leftarrow P \cup$ random_individual
6:	$Best \Leftarrow \varnothing$
7:	**repeat**
8:	**for** each $P_i \in P$ **do**
9:	**if** $Best = \varnothing$ or $K(P_i) > K(Best)$ **then**
10:	$Best \Leftarrow P_i$
11:	$Q \Leftarrow \{$the n best individuals from $P\}$
12:	**for** $(popsize - n)/2$
13:	$C_a, C_b \Leftarrow$ Crossover(Tournament(P), Tournament(P))
14:	$Q \Leftarrow Q \cup \{$Mutate(C_a), Mutate(C_b)$\}$
15:	$P \Leftarrow Q$
16:	**until** *Best* is ideal or time limit is reached
17:	**return** *Best*

Algorithm 2 - Genetic Algorithm with Elitism

Why use Genetic Algorithm?

- Uses a different methodology to create diversification than Simulated Annealing.
- Takes longer to process, but may get better and/or faster results.

Why use Elitism?

- Creates intensification in the best solutions.

Parameters:

- *pop size* = Size of the Population
- *elite amount* = Amount of best individuals kept each iteration
- *mutation prob* = Probability of each index getting mutation

Algorithm 2 - Genetic Algorithm with Elitism

Specific Processes:

- Tournament of 4 = Select 4 random individuals within the population and select the one with the best fitness
- One-Point Crossover = Defines one index as a crossing point, giving one side of the solution to a child e the rest to the other child.
- Mutation = Generates a random chance, based on mutation probability, of one value of the solution being changed for another value

Datasets - NORAD Two-Line Elements

As suggested in the reference paper, the NORAD TLE web-page[6] contains various satellites' information, which includes lists of debris described in a specific format[7].
Some of the useful information within each element are:

- Satellite Number
- Orbit Inclinations [Degrees]
- Motion [Revolutions per day]
- Eccentricity

[6] NORAD Two Line Elements. link to web-page
[7] NORAD Two-Line Element Set Format. link to web-page

Datasets - NORAD Two-Line Elements

Some of the available data in TLE format contain debris information and will be used as reference values to compute cost matrix.

Data used:

- FENGYUN 1C Debris[8]:
  - 2620 debris
- IRIDIUM 33 Debris[9]:
  - 314 debris
- COSMOS 2251 Debris[10]:
  - 1027 debris

[8] FENGYUN 1C Debris accessed at May 25th2018
[9] IRIDIUM 33 Debris accessed at May 25th2018
[10] COSMOS 2251 Debris accessed at May 25th2018

Parameters Evaluation

Simulated Annealing on COSMOS 2251 Debris:

- Best of 200 executions:

loop / *alpha*	**0.900**	0.950	0.990	0.999
20	152	151	146	151
50	**146**	154	148	150
100	151	153	149	154

- Mean of 200 executions:

loop / *alpha*	0.900	**0.950**	0.990	0.999
20	184.140	185.645	185.165	186.790
50	179.400	179.235	178.620	178.700
100	174.480	**174.110**	175.115	175.515

Parameters Evaluation

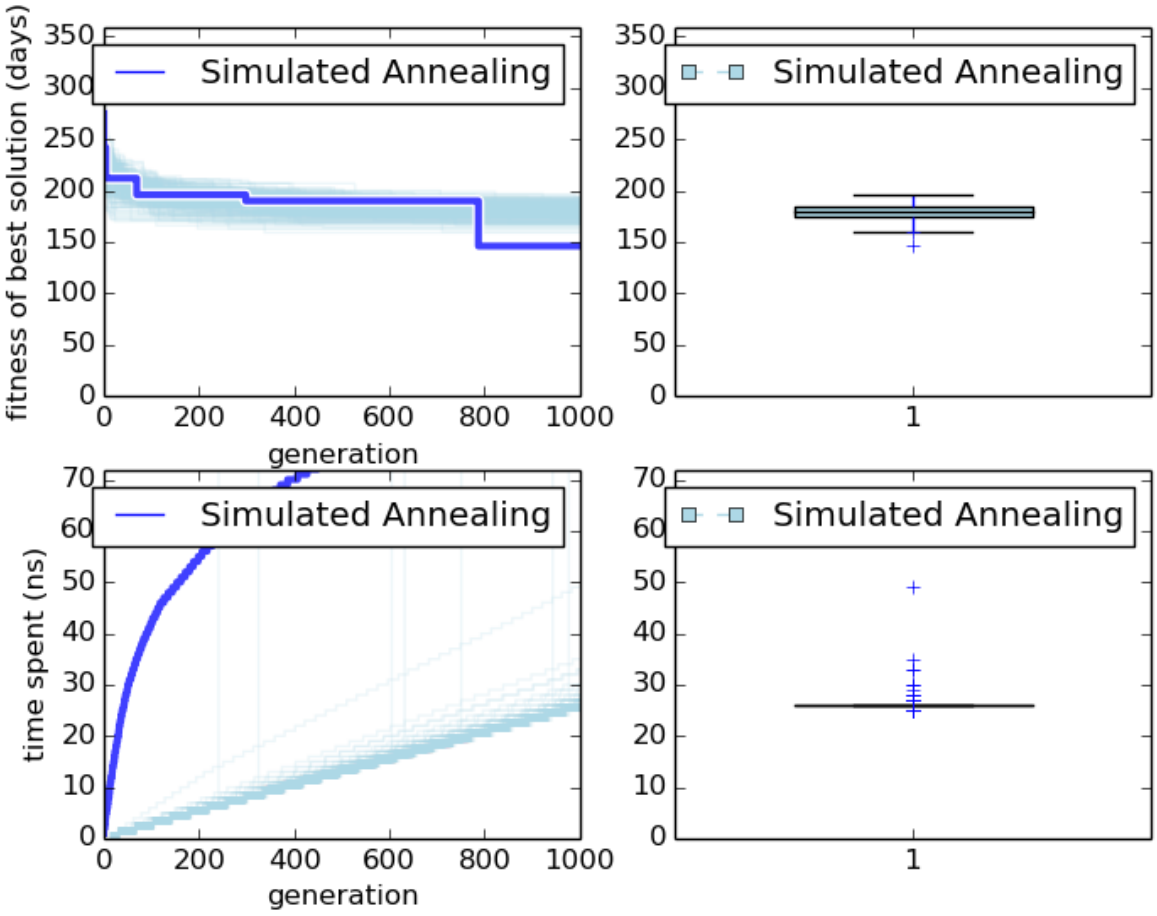

Figure: Best Solution after 200 executions (alpha 0.9, loop = 50.

Parameters Evaluation

Genetic Algorithm on COSMOS 2251 Debris:

- Best of 200 executions:

pop / *elite*	2	4	10	**16**
20	112	117	125	128
50	116	110	117	**94**
100	114	117	118	115

- Mean of 200 executions:

pop / *elite*	**2**	4	10	16
20	145.130	144.335	146.870	155.615
50	139.615	140.190	140.710	140.865
100	**137.755**	139.315	140.485	138.400

Parameters Evaluation

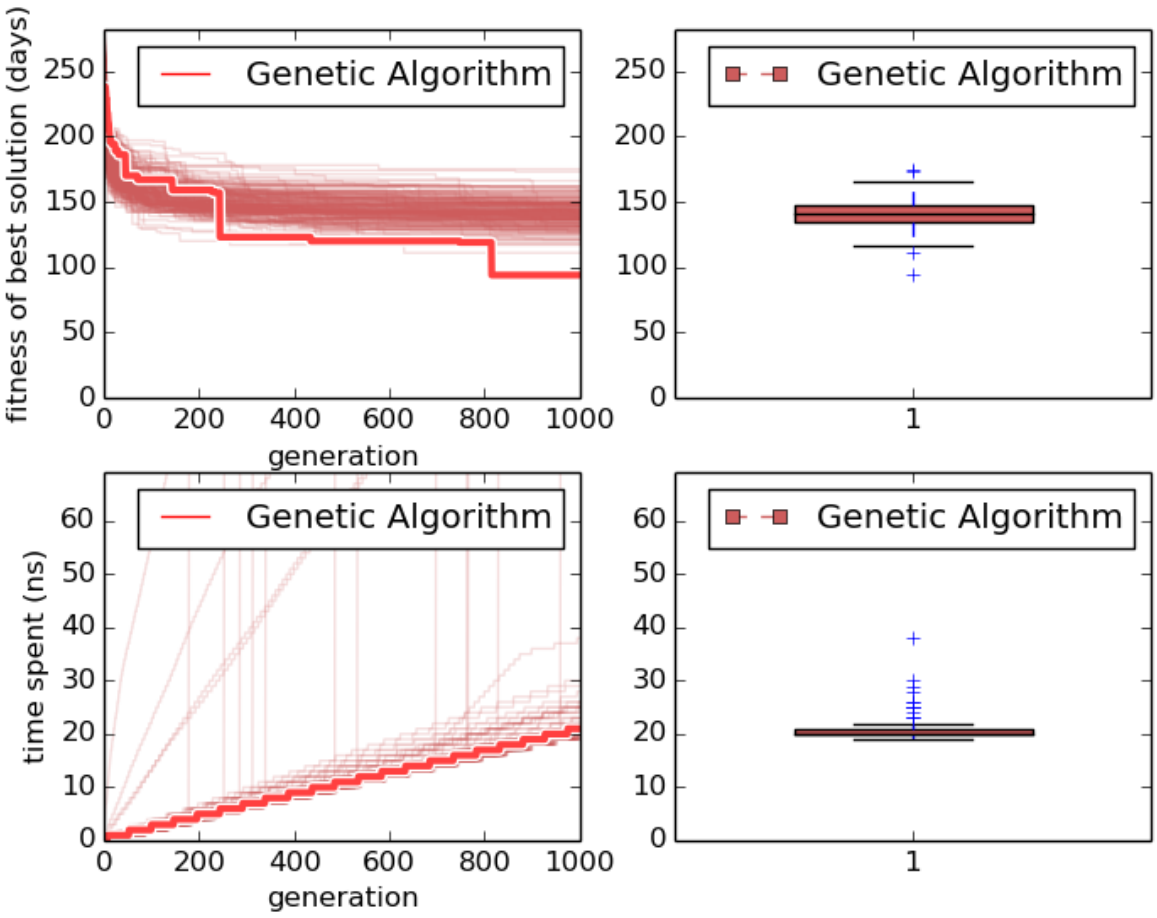

Figure: Best Solution after 200 executions (popsize = 100, elite = 10.

Algorithm Comparing

100 executions of each algorithm in COSMOS 2251 Debris:

- Fitness:

Algorithm (Parameters)	Best	Mean	Deviation
Simulated Annealing (Best)	159	179.320	**8.100**
Genetic Algorithm (Best)	**104**	140.660	11.001
Simulated Annealing (Mean)	156	175.770	6.971
Genetic Algorithm (Mean)	118	**138.900**	8.838

- Execution Time:

Algorithm (Parameters)	Best	Mean	Deviation
Simulated Annealing (Best)	**19**	**21.500**	**8.243**
Genetic Algorithm (Best)	**19**	42.210	114.822
Simulated Annealing (Mean)	38	42.210	11.584
Genetic Algorithm (Mean)	51	81.530	122.969

Algorithm Comparing - Parameters for Best Fitness

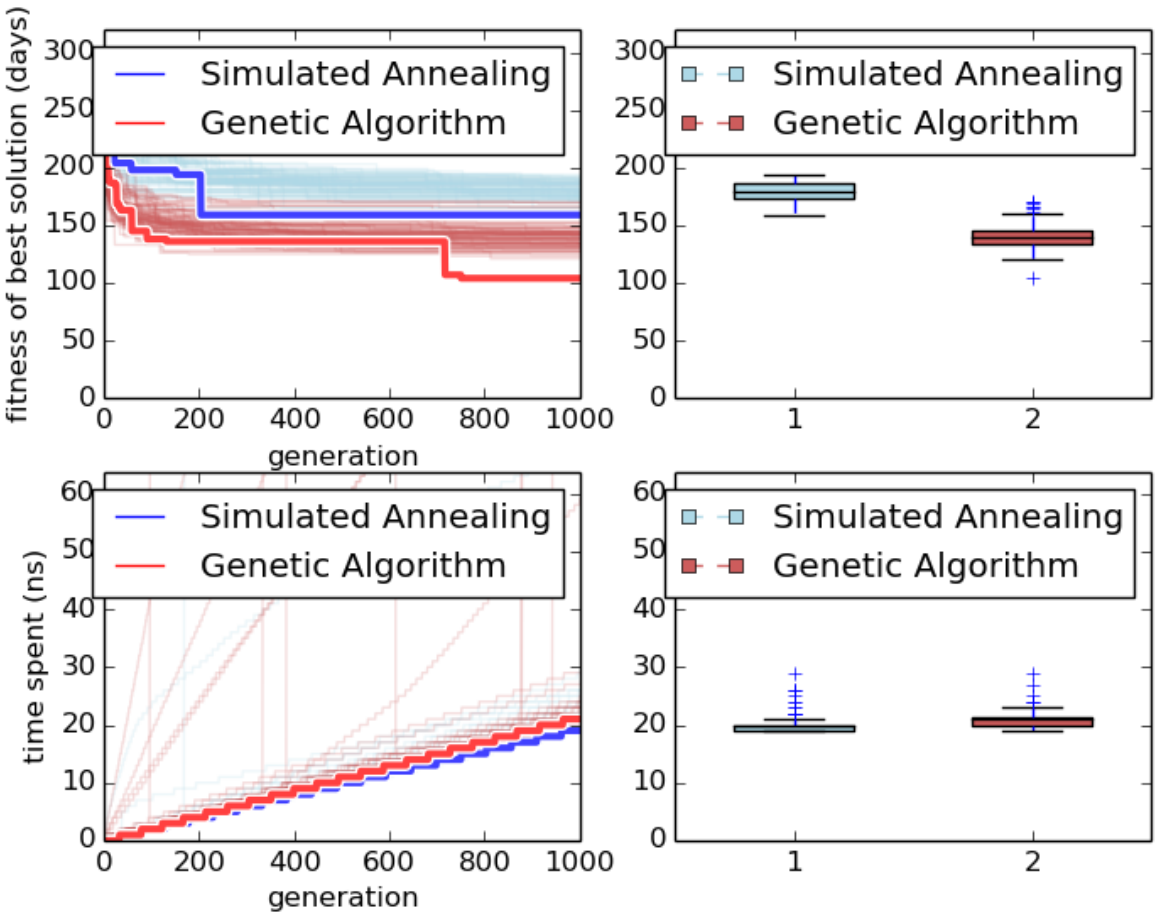

Figure: Comparing results using both algorithms with parameters that reached best fitness.

Algorithm Comparing - Parameters for Mean Fitness

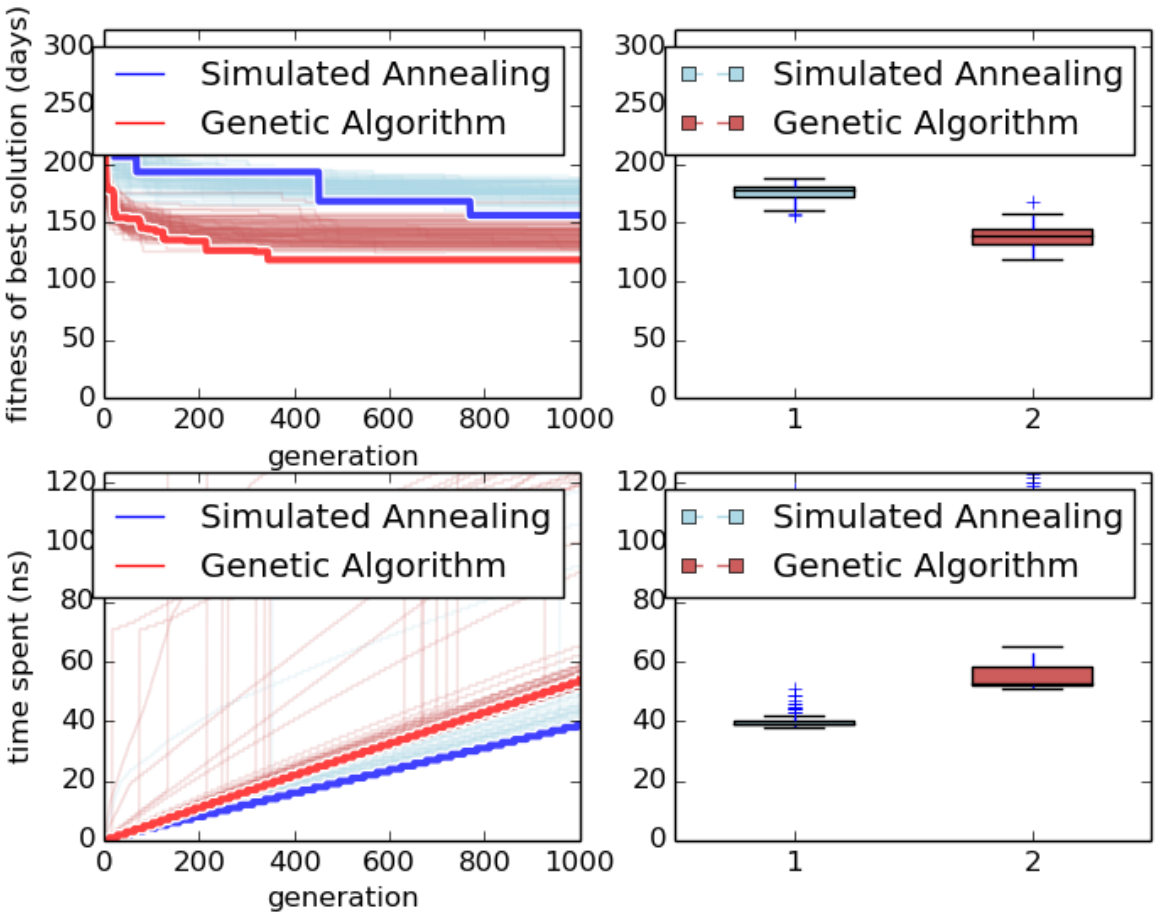

Figure: Comparing results using both algorithms with parameters that reached best mean fitness.

Results - IRIDIUM 33 Debris

Results for both algorithms in IRIDIUM 33 Debris:

- Best Solution:
  - Genetic Algorithm = [259,298,216,143,79]
- Fitness:

Algorithm	Best	Mean	Deviation
Simulated Annealing	139	163.260	**9.646**
Genetic Algorithm	**95**	**129.040**	12.514

- Execution Time:

Algorithm	Best	Mean	Deviation
Simulated Annealing	**17**	**29.680**	**12.288**
Genetic Algorithm	19	48.760	59.430

Results - IRIDIUM 33 Debris

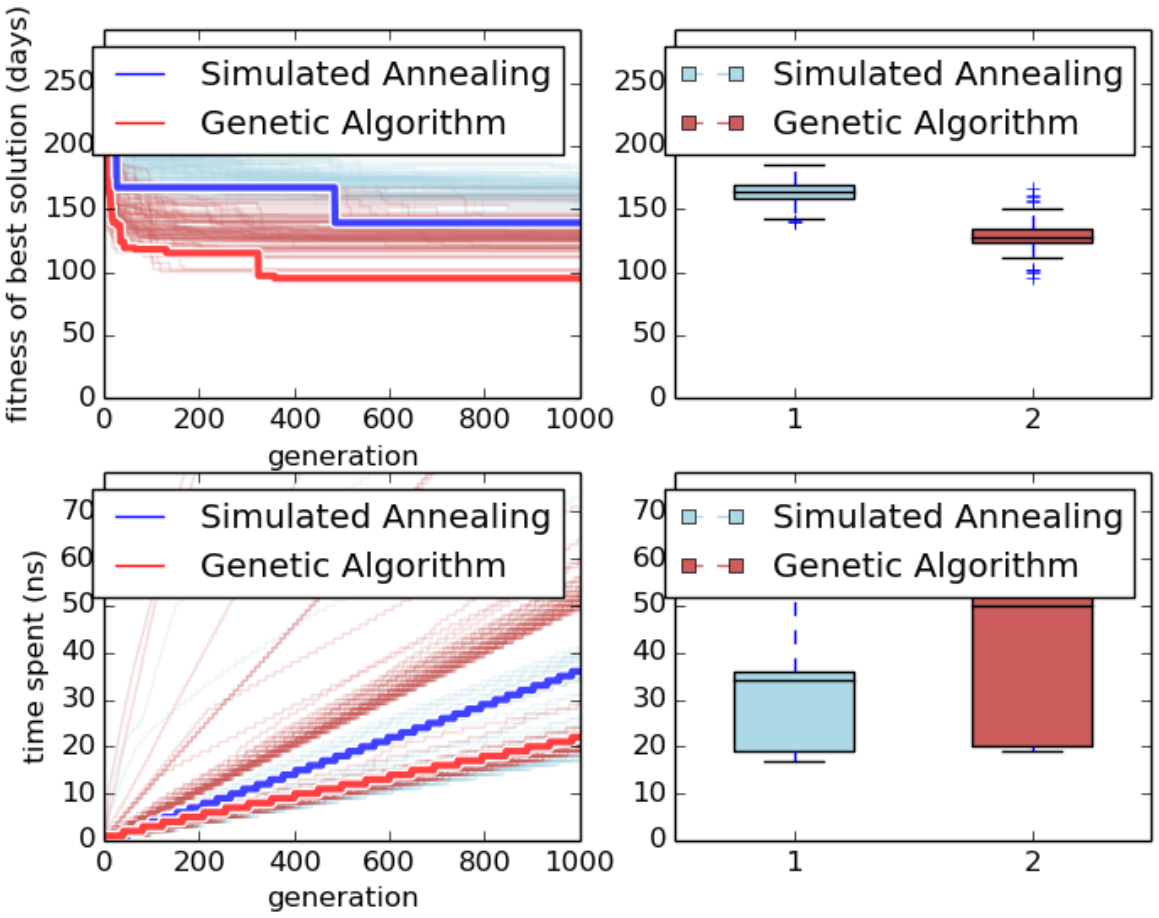

Figure: Comparing results using both algorithms in IRIDIUM 33 Debris.

Results - COSMOS 2251 Debris

Results for both algorithms in COSMOS 2251 Debris:

- Best Solution:
  - Genetic Algorithm = [326,683,562,190,532]
- Fitness:

Algorithm	Best	Mean	Deviation
Simulated Annealing	160	177.660	**6.892**
Genetic Algorithm	**116**	**140.570**	10.169

- Execution Time:

Algorithm	Best	Mean	Deviation
Simulated Annealing	21	**43.280**	**110.406**
Genetic Algorithm	**20**	60.120	125.277

Results - COSMOS 2251 Debris

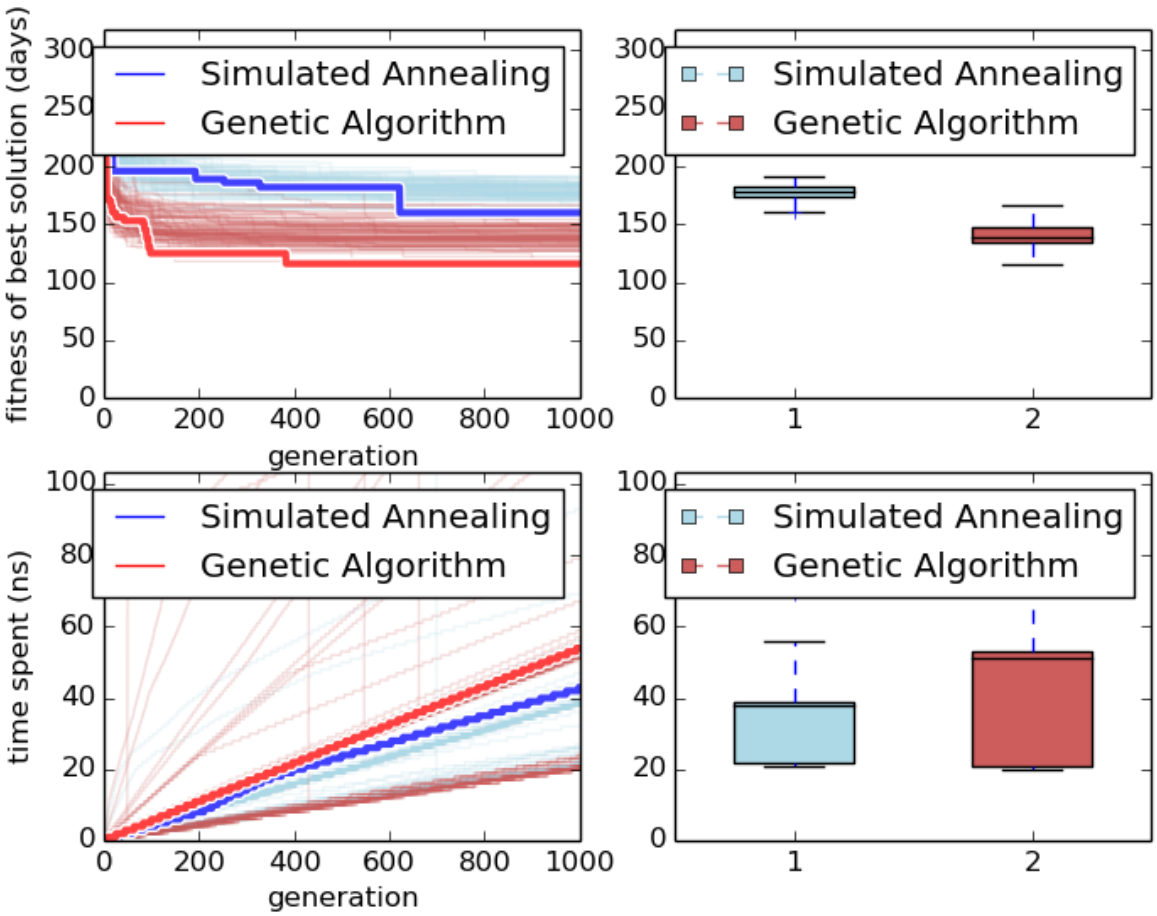

Figure: Comparing results using both algorithms in COSMOS 2251 Debris.

Questions?

Repository: `https://github.com/DracoScript/SDC-Optimization`