Global Optimization of MINLP by Evolutionary Algorithms

Martin Schlueter

Information Initiative Center
Hokkaido University, Japan

APMonitor Webinar (AIChE)

presented by
John Hedengren, PRISM Group, BYU

26^{th} Feb 2014

Outline

1. Optimization Problem
   - MINLP - Mixed Integer Nonlinear Programming
2. Evolutionary Algorithms
   - ACO - Ant Colony Optimization
   - Oracle Penalty Method
3. Software
   - MIDACO - Mixed Integer Distributed Ant Colony Optimizaton
4. Numercial Results
   - Comparison with arGA (Genetic Algorithm)
   - Comparison with BONMIN, COUENNE & MISQP
5. MINLP Space Applications
   - Multi-Stage Launch Vehicle (Boing Delta III)
   - Interplanetary Space Mission (NASA Galileo)
6. References
7. Conclusions

The Optimization Problem

MINLP

Mixed Integer Nonlinear Programming

The Optimization Problem

General MINLP problem:

$$\text{Minimize} \quad f(x,y) \qquad (x \in \mathbb{R}^{n_{con}},\ y \in \mathbb{Z}^{n_{int}},\ n_{con},\ n_{int} \in \mathbb{N})$$

$$\begin{aligned}
\text{subject to:} \quad g_i(x,y) &= 0, \quad i = 1, ..., m_e \in \mathbb{N} \\
g_i(x,y) &\geq 0, \quad i = m_e + 1, ..., m \in \mathbb{N} \\
x_l \leq x &\leq x_u \quad (x_l,\ x_u \in \mathbb{R}^{n_{con}}) \\
y_l \leq y &\leq y_u \quad (y_l,\ y_u \in \mathbb{N}^{n_{int}})
\end{aligned}$$

- No information on $f()$ or $g()$ available **[Blackbox]**

  $\longrightarrow$ non-convex, no gradients, stochastic noise

- Integers must be integers (no relaxation)

Evolutionary Algorithms

Evolutionary Algorithms

Evolutionary Algorithms

Key Features of Evolutionary Algorithms

1. Evolution: Random Mutations + Survival of the fittest
2. Goal: Find a good solution in reasonable time
3. GOOD: **[Black-Box]** Robust & (Very) Easy to use
4. BAD: No guarantee & Many Evaluation
5. **Very popular**

"optimization algorithm"	gets	1,500,000	Google hits
"evolutionary algorithm"	gets	730,000	Google hits
"numerical algorithm"	gets	390,000	Google hits
"deterministic algorithm"	gest	180,000	Google hits

Evolutionary Algorithms

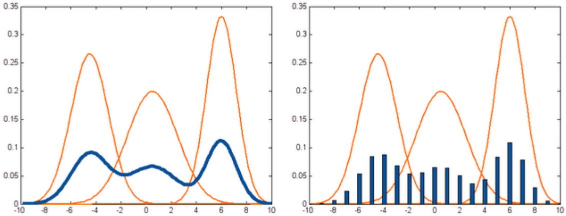
Evolutionary Algorithms for MINLP

1. (Very) young field ⟶ lot's of unexplored research opportunities
2. Very few software codes available
3. Hardly any comparison with deterministic MINLP algorithms

ACO - Ant Colony Optimization

ACO - Ant Colony Optimization

(or: Stochastic Gauss Approximation Algorithm)



ACO - Ant Colony Optimization

ACO in a Nutshell

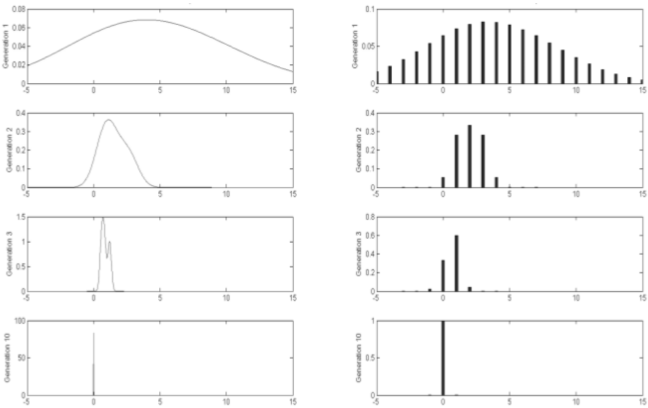
1. Terminology: Ant = Solution (x, y) & Fitness = Objective $f(x, y)$
2. Step 1: Randomly choose a number of P Ants $\longrightarrow$ Initial Solutions
3. Step 2: Select the number of K Ants with the best fitness
4. Step 3: Use Gauss-PDF (on K best Ants) to create P new Ants

   Repeat Step 2 & 3

5. P stands for *Population* size, P must be larger than K
6. K stands for *Kernel* size in multi-kernel Gauss PDF's
7. Every Step 2 is called a *Generation* of Ants
8. Step 3 can be **(massively)** **parallelized**

ACO - Ant Colony Optimization

ACO convergence of multi-kernel PDF's



PDF for continous x

(Solution $x = 0$)

PDF for discrete y

(Solution $y = 0$)

Oracle Penalty Method

Oracle Penalty Method

Oracle Penalty Method

The Oracle Penalty Method in a Nutshell

1. **Idea**: Help the algorithm by providing expert knowledge ⟶ Oracle

2. **Example:** Engineer has an application wich currently cost 1000$

   ACO algorithm finds in 1st Generation three solutions:

   Solution 1: f(x) = 2000$ (feasible)
   Solution 2: f(x) = 800$ (slightly infeasible)
   Solution 3: f(x) = 100$ (very infeasible)

   Which solution should be further investigated by ACO ?

   Static Penalty: Solution 1 is preferred
   Oracle Penalty: Solution 2 is preferred

Oracle Penalty Method

What if the expert **has no clue** about the (global) optimal f(x) value ?

$\longrightarrow$ Automatic Oracle Update (MIDACO Default)

ACO Run 0 with oracle $\Omega = \infty$ $\longrightarrow$ solution 1100$ (feasible)
ACO Run 1 with oracle $\Omega = 1100$ $\longrightarrow$ solution 900$ (feasible)
ACO Run 2 with oracle $\Omega = 900$ $\longrightarrow$ solution 855$ (feasible)
ACO Run 3 with oracle $\Omega = 855$ $\longrightarrow$ solution 850$ (feasible)
ACO Run 4 with oracle $\Omega = 850$ $\longrightarrow$ solution 848$ (infeasible)
ACO Run 5 with oracle $\Omega = 850$ $\longrightarrow$ solution 849$ (infeasible)

Overall solution 850$ found with an oracle Ω=855

The savest oracle is slightly above the global optimal f(x)

Careful: Underestimated oracles often lead to infeasible solutions

Oracle Penalty Method

Mathematical formulation of the penalty (merit) function $p(x)$, depending on the oracle parameter Ω

$$p(x) = \begin{cases} \alpha \cdot |f(x) - \Omega| + (1-\alpha) \cdot res(x) & , \text{ if } \quad f(x) > \Omega \quad \text{or} \quad res(x) > 0 \\ -|f(x) - \Omega| & , \text{ if } \quad f(x) \leq \Omega \quad \text{and} \quad res(x) = 0 \end{cases}$$

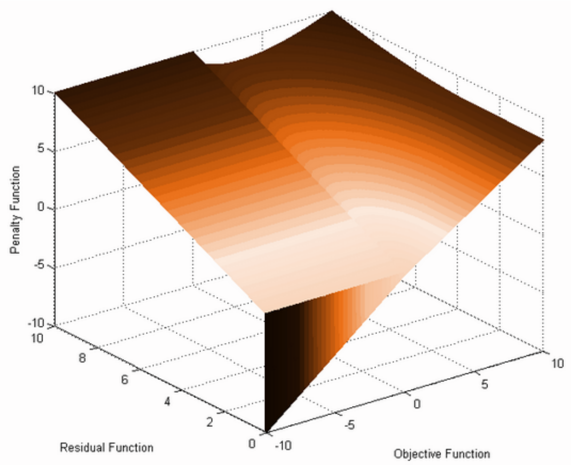
where α is given by:

$$\alpha = \begin{cases} \dfrac{|f(x)-\Omega| \cdot \frac{6\sqrt{3}-2}{6\sqrt{3}} - res(x)}{|f(x)-\Omega| - res(x)} & , \text{ if } \quad f(x) > \Omega \quad \text{and} \quad res(x) < \frac{|f(x)-\Omega|}{3} \\ 1 - \dfrac{1}{2\sqrt{\frac{|f(x)-\Omega|}{res(x)}}} & , \text{ if } \quad f(x) > \Omega \quad \text{and} \quad \frac{|f(x)-\Omega|}{3} \leq res(x) \leq |f(x) - \Omega| \\ \frac{1}{2}\sqrt{\frac{|f(x)-\Omega|}{res(x)}} & , \text{ if } \quad f(x) > \Omega \quad \text{and} \quad res(x) > |f(x) - \Omega| \\ 0 & , \text{ if } \quad f(x) \leq \Omega \end{cases}$$

Note that $res(x)$ is the constraint violation, e.g. L_1-Norm

Oracle Penalty Method

Graphical illustration of the oracle penalty function for $\Omega = 0$)



The Optimization Software

Mixed Integer Distributed Ant Colony Optimization

The Optimization Software

Key features of MIDACO:

1. Written entirely from scratch in F77 (7+ Years of Development)
2. No sub-solvers or external libraries (e.g. LAPACK or BOOST)
3. Written in (blind) reverse communication ⟶ portable everywhere
4. Language: Excel/VBA, Matlab, Octave, Python, C++, Fortran, ...
5. MIDACO 4.0 is designed for up to **1000 variables**
6. Suitable for **massive parallelization** (Multi-Core, HPC & GPGPU)
7. **Very user friendly** to compile, execute and operate

The Optimization Software

Some (academic) applications of MIDACO

Area	Application	Author(s)
Space	Interplanetary Space Mission (NASA Galileo)	Schlueter et al.
Space	Space Launch Vehicle (Boeing Delta III)	Schlueter et al.
Space	Thermal Insulation System (Heat Shield)	Schlueter et al.
Space	Satellite Constellation	Takano, Marchand
Electr-Eng	Control of Cogeneration Systems	Pandurangan
Robotics	Optimal Camera Placement	Hänel et al.
Climate	Nonlinear Model Predictive Control	Booij, Sijs, Fransman
Finance	Distance-to-Default	Allugundu, Kumar
Finance	Application of Sales Forecasting	Comas
Bio-Tech	Parameter optimization in Bio-Technology	Rehberg et al.
Telecom	Cooperative Wireless Networks	Baidas, MacKenzie
Navy	Structural Optimization of Submarine Hulls	Wong

List of commercial applications available on request.

The Optimization Software

Please visit: http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html

GLOBAL TRAJECTORY OPTIMISATION PROBLEMS DATABASE

advanced concepts team
informatics and applied maths
esa

Benchmark Name	Variables	Constraints	Number of submissions	Time between first and last submission
Cassini1	6	4	3	6 Month
GTOC1*	8	6	2	13 Month
Messenger (reduced)	18	0	3	11 Month
Messenger (full)*	26	0	8	55 Month
Cassini2*	22	0	7	14 Month
Rosetta	22	0	7	6 Month
Sagas	12	2	1	-

* **Best known solution found by MIDACO**

The Optimization Software

MIDACO holds **1st** and **2nd** record solution on Messenger (Full) [Hardest ESA-GTOP problem]

OBJECTIVE FUNCTION (KM/S)	SOLUTION VECTOR	CREDITS:	DATE:
6.943	N/A	M. Schlueter, J. Fiala, M. Gerdts, University of Birmingham (found by MIDACO solver)	19/06/2009
6.404	N/A	G. Stracquadanio, A. La Ferla, G. Nicosia, University of Catania (Found by SAGES Self-Adaptive- Gaussian Evolutionary Strategy)	17/11/2009
6.047	N/A	M. Schlueter, University of Birmingham, M. Gerdts, University of Wuerzburg, M. Munetomo and K. Akama, Hokkaido University, S. Erb and G. Ortega, ESTEC/TEC-ECM (found by MIDACO solver)	30/11/2009
4.254	N/A	F. Biscani and D. Izzo, ESTEC Advanced Concepts Team. Found using PaGMO	01/12/2009
2.970	CLICK HERE	G. Stracquadanio, Dept of Biomedical Engineering, Johns Hopkins University, A. La Ferla, G. Nicosia, University of Catania (Found by SAGES Self-Adaptive-Gaussian Evolutionary Strategy)	28/02/2011
2.113	CLICK HERE	G. Stracquadanio, Dept of Biomedical Engineering, Johns Hopkins University, A. La Ferla, G. Nicosia, University of Catania (Found by SAGES Self-Adaptive-Gaussian Evolutionary Strategy)	10/04/2012
2.104	CLICK HERE	M. Schlueter, M. Munetomo (found by MIDACO solver)	17/10/2013
1.983	CLICK HERE	M. Schlueter, (found by MIDACO solver)	11/02/2014

Record Nr. 1 (Feb 2014)	Record Nr. 2 (Nov 2013)
1.983 km/sec	**2.104** km/sec
F(x) = 1.983000935817071	F(x) = 2.104185170661480
x[0] = 2037.777752194555205;	x[0] = 2060.627272281109072;
x[1] = 4.036037772605158;	x[1] = 4.042601735668291;
x[2] = 0.555461305960588;	x[2] = 0.440387114371649;
x[3] = 0.636411860398641;	x[3] = 0.653458177621111;
x[4] = 451.456424896320755;	x[4] = 428.903525341671866;
x[5] = 224.694309184162989;	x[5] = 224.687235869007964;
x[6] = 221.881079005111076;	x[6] = 221.385427446748679;
x[7] = 265.245236722557934;	x[7] = 266.124367319569956;
x[8] = 358.326887762935030;	x[8] = 358.048599982140672;
x[9] = 534.184307028764806;	x[9] = 444.429427422362778;
x[10] = 0.599458381121838;	x[10] = 0.581561467686441;
x[11] = 0.739157714128318;	x[11] = 0.821640755039470;
x[12] = 0.719150322680764;	x[12] = 0.698772357707937;
x[13] = 0.759748648439657;	x[13] = 0.720609016544215;
x[14] = 0.828459261980907;	x[14] = 0.829340143768712;
x[15] = 0.902545986525688;	x[15] = 0.875166415983967;
x[16] = 1.424681465904589;	x[16] = 1.576183861868870;
x[17] = 1.100000319782133;	x[17] = 1.100003220464050;
x[18] = 1.051121973230687;	x[18] = 1.052869945803204;
x[19] = 1.150030195232458;	x[19] = 1.050000430115585;
x[20] = 1.050000606119374;	x[20] = 1.477180737136582;
x[21] = 2.820044490192342;	x[21] = 2.786201469971995;
x[22] = 1.514855471162370;	x[22] = 1.603649010967501;
x[23] = 2.589799685625540;	x[23] = 2.622074959673106;
x[24] = 1.756472273431577;	x[24] = 1.571933956929996;
x[25] = 1.561537932872780;	x[25] = 1.606318012513329;

8 Solution submission over a period of **4.5 Years** ⟶ **Very Hard**

Numerical Results

MIDACO comparison with arGA (Munawar et al.)

Numerical Results

Test Setup

1. Language: C/C++ (i7 CPU Q920@2.67GHz)
2. Number of problems: 8 (maximal 17 variables) $\longrightarrow$ **Toy Problems**
3. Number of test runs: 30
4. Starting point: Lower bounds
5. Stopping criteria: Optimum reached within 1% or 100 Seconds
6. Accuracy for constraints: 0.01
7. MIDACO Version: 4.0

Numerical Results

Comparison of arGA and MIDACO on 8 toy problems

	arGA			MIDACO			Speed-Up	
Nr.	Optimal	Eval	Time	Optimal	Eval	Time	Eval	Time
1	30	1976	0.21	30	1803	0.0011	1.09	190
2	30	11301	1.38	30	9593	0.0068	1.17	202
3	30	24253	2.87	30	21047	0.0171	1.15	167
4	21	14381	1.48	30	2521	0.0018	5.70	822
5	25	41626	5.92	30	26367	0.0248	1.57	238
6	17	26092	3.21	30	483	0.0004	54.02	8025
7	16	92646	10.86	30	101797	0.1934	0.91	56
8	16	73581	8.55	30	6255	0.0044	11.76	1943
	77%			**100%**			**9.67**	**1455**

⟶ arGA solves only the first 3 problems robustly

⟶ MIDACO solves **100%** about **1500 times** faster than arGA

Numerical Results

Conclusions of Comparison of arGA and MIDACO:

1. The arGA development took around **1 Year**
2. The MIDACO development took around **7 Years**
3. Both are evolutionary algorithms
4. Both softwares claim to be *"sophisticated"*

MIDACO is **significantly** stronger than arGA in regard to robustness,

algorithmic efficiency and overall cpu-time performance.

⟶ There is a great variety regarding the **quality** of implemenations of evolutionary algorithms

Numerical Results

MIDACO comparison with MISQP

MIDACO comparison with MISQP

Test Setup

1. Language: Fortran (i7 CPU Q920@2.67GHz)
2. Number of problems: 100 (max 100 variables, 66 GAMS instances)
3. Number of MISQP test runs: 1
4. Number of MIDACO test runs: 10
5. MISQP Starting point: Pre-defined X_0 (mostly GAMS default)
6. MIDACO Starting point: Lower bounds
7. Stopping criteria: Optimum reached within 1% or 300 Seconds
8. Accuracy for constraints: 0.0001
9. MIDACO Version: 3.0

MIDACO comparison with MISQP

MISQP

Algorithm	***Optimal***	*Feasible*	$Eval_{mean}$	$Time_{mean}$
MISQP	**89**	100	500	0.39
MISQP/bmod	**71**	100	340	0.20
MISQP/fwd	**81**	100	396	0.11
MISQP/rst0	**69**	99	241	0.14
MISQPOA	**91**	100	1,093	0.65
MISQPN	**74**	98	1,139	0.17
MINLPB4/bin	**92**	100	1,787	30.91
MINLPB4/int	**88**	94	218,881	4.11

Maximal
92 Optimal
(30.91 Sec)

MIDACO

Seed	***Optimal***	*Feasible*	$Eval_{mean}$	$Time_{mean}$
0	**96**	99	1,656,979	4.54
1	**96**	99	3,223,015	9.01
2	**96**	98	1,673,873	4.20
3	**97**	98	2,235,463	6.53
4	**96**	98	2,054,099	6.08
5	**97**	99	1,485,525	4.82
6	**95**	99	1,641,648	4.07
7	**97**	99	1,724,627	5.90
8	**95**	99	1,120,204	2.77
9	**96**	98	2,313,829	7.93

Maximal
97 Optimal
(6.53 Sec)

Numerical Results

MIDACO comparison with BONMIN & COUENNE

MIDACO comparison with BONMIN & COUENNE

Test Setup

1. BONMIN, COUENNE Language: GAMS (Free Gradients?)
2. MIDACO Language: Fortran
3. Number of problems: 66 (max 48 variables, GAMS MINLP's)
4. Number of BONMIN, COUENNE test runs: 1
5. Number of MIDACO test runs: 10
6. BONMIN, COUENNE Starting point: Pre-defined X_0
7. MIDACO Starting point: Lower bounds
8. BONMIN, COUENNE Stopping criteria: Autostop or 300 Seconds
9. MIDACO Stopping criteria: Autostop (Value 50) or 300 Seconds
10. Accuracy for constraints: 0.0001
11. MIDACO Version: 3.0 (all solvers run on i7 CPU Q820@1.73GHz)

MIDACO comparison with BONMIN & COUENNE

Performance of BONMIN, COUENNE & MIDACO on 66 GAMS MINLP's

Solver	***Optimal***	*Feasible*	$Time_{aver}$	$Time_{total}$
BONMIN	**49**	64	17.99	1187.62
COUENNE	**48**	64	40.36	2664.31
MIDACO	**51 ∼ 62**	64 ∼ 65	31.41	2072.73

BONMIN: 49 Optimal
COUENNE: 48 Optimal
MIDACO: 62 Optimal

Numerical Results

Conclusions of Comparison with BONMIN, COUENNE & MISQP

1. Test problems were in favor of BONMIN, COUENNE & MISQP
2. Starting points were in favor of BONMIN, COUENNE & MISQP
3. Environment was in favor of BONMIN & COUENNE

MIDACO can **outperform** BONMIN, COUENNE & MISQP on small to mid-scale MINLP's in regard to reaching the **global optimum** (fast).

MIDACO cpu-runtime performance is competetive.

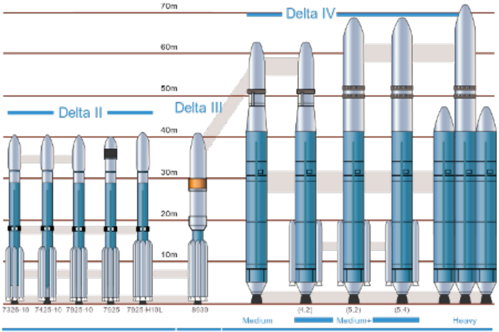
MIDACO needs much more function evaluation. ⟶ **parallel**

MINLP Space Applications

MINLP Space Applications

Ascent of Multi-Stage Launch Vehicle (Delta III)

Ascent of Multi-Stage Launch Vehicle



Boing Delta Rocket Family

Ascent of Multi-Stage Launch Vehicle (Delta III)

MIDACO has been used to optimize the ascent of a multi-stage launch vehicle. The model was based on a Delta III Space Rocket (Boeing) and the formulation by V. Rao (GPOPS) was considered. The ascent of the vehicle is formulated as optimal control problem of a constrained system of (discretized) ordinary differential equations (ODE's). The number of active strap-on boosters in Stage 1 and Stage 2 is considered to be an integer variable, as well as their manufactor type.

MINLP problem specifications:

1. 128 decision variables
2. 3 integer variables
3. 127 constraints
4. 5 equality constraints

Ascent of Multi-Stage Launch Vehicle (Delta III)

Integer Extension

Formulating the type and number of strap-on boosters as variable.
5 Different Booster types. Up to 9 active booster in first stage.

Table 5: Enumeration over all (feasible) booster configurations with $B_1 \geq 6$

Booster-Config.				Booster-Config.				Booster-Config.				Booster-Config.			
B_1	T_1	T_2	Best known $f(x,y)$	B_1	T_1	T_2	Best known $f(x,y)$	B_1	T_1	T_2	Best known $f(x,y)$	B_1	T_1	T_2	Best known $f(x,y)$
6	1	1	-6685.71	8	1	1	-6848.21	7	1	1	-6789.90	9	1	-	-6855.30
6	1	2	-6808.53	8	1	2	-6900.99	7	1	2	-6883.25	9	2	-	-7324.23
6	1	3	-6884.45	8	1	3	-6935.36	7	1	3	-6942.60	9	3	-	**-7599.88**
6	1	4	-6955.92	8	1	4	-6969.11	7	1	4	-6999.74				
6	1	5	-7055.32	8	1	5	-7018.56	7	1	5	-7081.49				
6	2	1	-7075.93	8	2	1	-7297.53	7	2	1	-7213.85				
6	2	2	-7195.10	8	2	2	-7228.32	7	2	2	-7303.58				
6	2	3	-7269.14	8	2	3	-7381.77	7	2	3	-7360.66				
6	2	4	-7339.15	8	2	4	-7414.42	7	2	4	-7415.64				
6	3	1	-7315.13	8	2	5	-7321.22	7	2	5	-7494.50				
6	3	2	-7431.81	8	3	1	**-7565.08**	7	3	1	-7271.36				
6	*3*	*3*	*-7504.48*	8	3	2	**-7614.97**	7	3	2	**-7556.82**				
6	4	1	**-7539.17**	8	3	3	**-7647.50**	7	3	3	**-7612.70**				

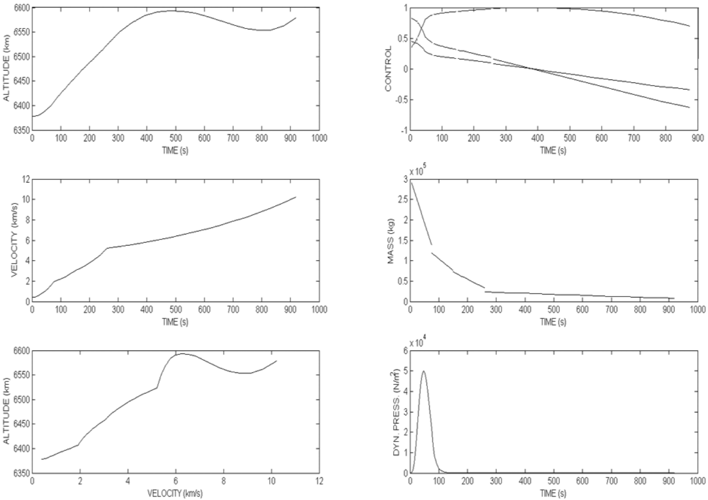
Overall best configuration: $y = \{8,3,3\}$, $f(x,y) = -7647.5(kg)$

Ascent of Multi-Stage Launch Vehicle (Delta III)

Table 7: 30 runs by MIDACO (max time = 7200) + SQP (max iter=1000)

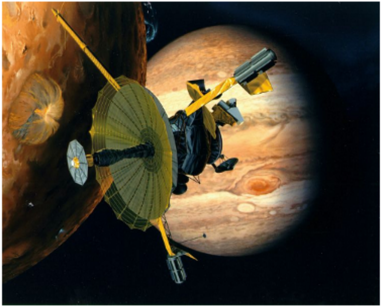
	Booster-Config.			SQP			MIDACO			
Run	B_1	T_1	T_2	$f(x,y)$	Eval	Time	$f(x,y)$	Eval	Time	
1	9	3	1	-7599.88	357908	317.7	-7419.65	3455790	7200.0	
2	9	3	1	-7599.88	353114	315.1	-7449.22	3450447	7200.0	
3	8	3	3	-7647.50	363366	321.1	-7502.77	3443609	7200.0	<- Optimal Solution reached
4	9	3	1	-7599.88	267022	236.8	-7419.91	3449060	7200.0	
5	9	3	5	-7599.88	309848	274.6	-7418.63	3460976	7200.0	
6	9	3	1	-7599.88	173384	153.8	-7436.00	3472466	7200.0	
7	9	3	1	-7599.88	346444	307.3	-7555.53	3456612	7200.0	
8	9	3	4	-7599.88	265638	234.8	-7369.10	3457577	7200.0	
9	7	3	3	-7567.75	6713	6.4	-7565.33	3445493	7200.0	
10	9	3	4	-7599.88	284148	254.1	-7524.85	3445318	7200.0	
11	8	3	3	-7524.57	7379	7.1	-7519.89	3447985	7200.0	
12	8	3	3	-7647.50	354988	313.6	-7481.90	3459946	7200.0	<- Optimal Solution reached
13	9	3	1	-7599.88	270324	240.1	-7444.49	3453002	7200.0	
14	8	3	3	-7647.50	363938	322.9	-7479.16	3451839	7200.0	<- Optimal Solution reached
15	9	3	5	-7599.88	266138	235.9	-7500.50	3464034	7200.0	
16	9	3	5	-7599.88	301198	266.8	-7519.93	3481049	7200.0	
17	9	3	3	-7599.88	344342	307.8	-7456.35	3450507	7200.0	
18	9	3	1	-7599.88	273766	242.3	-7528.82	3454741	7200.0	
19	9	3	1	-7599.88	298972	267.0	-7527.04	3458943	7200.0	
20	9	3	4	-7599.88	324916	290.1	-7431.08	3468826	7200.0	
21	9	3	5	-7599.88	355510	317.4	-7498.23	3487475	7200.0	
22	9	3	5	-7599.88	341446	322.4	-7430.29	3042720	7200.0	
23	8	3	3	-7647.43	309588	370.1	-7536.62	2879186	7200.0	<- Optimal Solution reached
24	9	3	5	-7599.88	349166	425.9	-7460.48	2435917	7200.0	
25	8	3	3	-7513.12	7360	9.2	-7505.67	2568537	7200.0	
26	6	4	1	-7539.17	361342	332.9	-7434.57	2871096	7200.0	
27	9	3	5	-7599.88	313390	313.4	-7348.84	3060132	7200.0	
28	9	3	3	-7599.88	263188	347.4	-7475.90	3150988	7200.0	
29	8	3	3	-7647.50	355292	321.1	-7470.97	2873982	7200.0	<- Optimal Solution reached
30	8	3	3	-7647.50	365252	327.5	-7491.21	3336049	7200.0	<- Optimal Solution reached

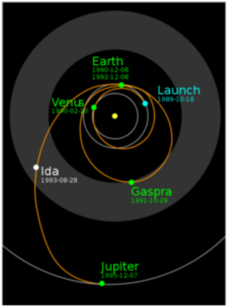
Ascent of Multi-Stage Launch Vehicle (Delta III)



Interplanetary Space Trajectory (MGA-DSM-MINLP)

Interplanetary Space Trajectory (MGA-DSM-MINLP)

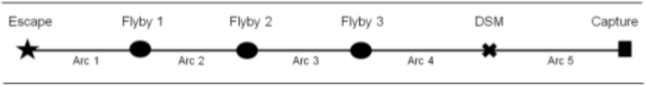




NASA's Galileo Mission (launched 1989)

Interplanetary Space Trajectory (MGA-DSM-MINLP)

Mission Layout (MGA-DSM)



Possible integer choices for Fly-By Planets:

Number	Planet
1	Mercury
2	Venus
3	Earth
4	Mars
5	Jupiter
6	Saturn
7	Uranus
8	Neptune
9	Pluto

MINLP: 21 Variables (3 Integer) & 12 Constraints

Interplanetary Space Trajectory (MGA-DSM-MINLP)

Table 9: Optimization variables x (continuous) and y (integer) with bounds

Variable	Description	Lower Bound	Upper Bound
continuous			
x_1	Launch Date	0 (01 Jan. 1989)	730 (31 Dec. 1990)
x_2	Duration of Arc 1	0 (days)	200 (days)
x_3	Duration of Arc 2	0 (days)	400 (days)
x_4	Duration of Arc 3	0 (days)	800 (days)
x_5	Duration of Arc 4	0 (days)	100 (days)
x_6	Duration of Arc 5	0 (days)	1200 (days)
x_7	Thrust Escape (X direction)	-6000.0 (m/sec)	6000.0 (m/sec)
x_8	Thrust Escape (Y direction)	-6000.0 (m/sec)	6000.0 (m/sec)
x_9	Thrust Escape (Z direction)	-3000.0 (m/sec)	3000.0 (m/sec)
x_{10}	Thrust Capture (X direction)	-6000.0 (m/sec)	6000.0 (m/sec)
x_{11}	Thrust Capture (Y direction)	-6000.0 (m/sec)	6000.0 (m/sec)
x_{12}	Thrust Capture (Z direction)	-3000.0 (m/sec)	3000.0 (m/sec)
x_{13}	Thrust DSM (X direction)	-1000.0 (m/sec)	1000.0 (m/sec)
x_{14}	Thrust DSM (Y direction)	-1000.0 (m/sec)	1000.0 (m/sec)
x_{15}	Thrust DSM (Z direction)	-500.0 (m/sec)	500.0 (m/sec)
x_{16}	Altitude Flyby 1	0.00 ($\sim Alt_{min}$)	1.00 ($\sim Alt_{max}$)
x_{17}	Altitude Flyby 2	0.00 ($\sim Alt_{min}$)	1.00 ($\sim Alt_{max}$)
x_{18}	Altitude Flyby 3	0.00 ($\sim Alt_{min}$)	1.00 ($\sim Alt_{max}$)
integer			
y_1	Planet Flyby 1	1 (Mercury)	9 (Pluto)
y_2	Planet Flyby 2	1 (Mercury)	9 (Pluto)
y_3	Planet Flyby 3	1 (Mercury)	9 (Pluto)

Interplanetary Space Trajectory (MGA-DSM-MINLP)

Table 13: 10 test runs by MIDACO on mission model with 3% sphere of action

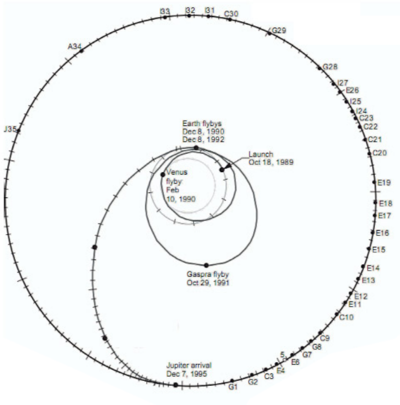
Run	Launch	ΔV	Duration	FlyBy 1	FlyBy 2	FlyBy 3
1	6 Nov. 1989	2553	5.88	Venus	Earth	Earth
2	30 Nov. 1989	3310	4.83	Venus	Earth	Earth
3	30 Nov. 1989	3218	4.88	Venus	Earth	Earth
4	10 Jul. 1989	3390	3.97	Venus	Earth	Mars
5	20 Nov. 1989	2890	4.77	Venus	Earth	Earth
6	23 May 1989	2759	5.35	Earth	Venus	Earth
7	21 Mar 1989	*infeasible*	4.85	Earth	Earth	Mars
8	13 Apr. 1989	3290	4.54	Earth	Venus	Earth
9	30 Nov. 1989	3289	4.79	Venus	Earth	Earth
10	16 Sep. 1989	2684	6.10	Venus	Earth	Earth

Table 14: Comparison between original Galileo and MIDACO Missions

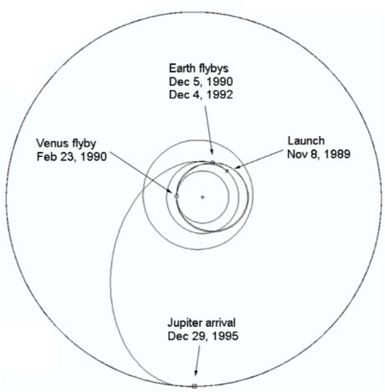
	Galileo Mission	Mission1 refine 0.5 %	Mission4 refine 0.5 %
Launch	18 Oct. 1989	8 Nov. 1989	6 Jul. 1989
Duration	6.14 Years	6.14 Years	4.15 Years
ΔV	*unknown*	3,350 m/sec	5,177 m/sec
1st Flyby			
Planet	Venus	Venus	Venus
Date	10 Feb. 1990	23 Feb. 1990	21 Jan. 1990
Altitude	16,000km	28,901km	3,013km
2nd Flyby			
Planet	Earth	Earth	Earth
Date	8 Dec. 1990	5 Dec. 1990	4 Sep. 1990
Altitude	960km	473,191km	1,754km
3rd Flyby			
Planet	Earth	Earth	Mars
Date	8 Dec. 1992	4 Dec. 1992	31 Dec. 1990
Altitude	303km	300km	39km

Interplanetary Space Trajectory (MGA-DSM-MINLP)

Comparison of original Galileo Trajectory and MIDACO Mission1



Galileo



MIDACO

Literature References (Preprints freely available @ MIDACO Homepage)

[1] Schlueter M., Gerdts M., Rueckmann J.J.:
A Numerical Study of MIDACO on 100 MINLP Benchmarks.
Optimization (Taylor & Francis), Vol 61, Issue 7, Pages 873-900 (2012)

[2] Schlueter M., Gerdts M.:
The Oracle Penalty Method.
Journal of Global Optimization (Springer), Vol. 47, Issue 2, Pages 293-325 (2010)

[3] Schlueter M., Erb S., Gerdts M., Kemble S., Rueckmann J.J.:
MIDACO on MINLP Space Applications.
Advances in Space Research (Elsevier), Vol 51, Issue 7, Pages 1116-1131 (2013)

[4] Schlueter M., Munetomo M.:
Parallelization Strategies for Evolutionary Algorithms for MINLP.
Proc. Cong. Evolutionary Computation (IEEE-CEC), Pages 635-641 (2013)

[5] Schlueter M.:
MIDACO Software Performance on Interplanetary Trajectory Benchmarks.
Advances in Space Research (Elsevier), *submitted* (2014)

Conclusions

With MIDACO as a representative of evolutionary algorithms, it was shown:

1. Evo. alg. for MINLP is a new field, worth of investigation
2. Evo. alg. can often find the global optimum fast(er)
3. Evo. alg. need many function evaluation $\longrightarrow$ **Not if parallelized!**
4. MIDACO is (probably) the **strongest** evolutionary MINLP software
5. MIDACO holds **1st** (and 2nd) record on **hardest ESA benchmark**.

Thank you for your attention!

Special Thanks:

John Hedengren, APMonitor & AIChE Cast Division