Numerical optimization of ESA's Messenger space mission benchmark

Martin Schlueter*, Mohamed Wahib†, and Masaharu Munetomo*

*Information Initiative Center, Hokkaido University, Sapporo 060-0811, Japan.
†RIKEN Advanced Institute for Computational Science, 7-1-26
Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo, 650-0047, Japan.
{schlueter@midaco-solver.com,
mohamed.attia@riken.jp,
munetomo@iic.hokudai.ac.jp}

Abstract. The design and optimization of interplanetary space mission trajectories is known to be a difficult challenge. The trajectory of the Messenger mission (launched by NASA in 2004) is one of the most complex ones ever created. The European Space Agency (ESA) makes available a numerical optimization benchmark which resembles an accurate model of Messengers full mission trajectory. This contribution presents an optimization approach which is capable to (robustly) solve ESA's Messenger full mission benchmark to its putative global solution within 24 hours run time on a moderate sized computer cluster. The considered algorithm, named MXHPC, is a parallelization framework for the MIDACO optimization algorithm which is an evolutionary method particularly suited for space trajectory design. The presented results demonstrate the effectiveness of evolutionary computing for complex real-world problems which have been previously considered intractable.

Keywords: Interplanetary Space Trajectory; Optimization; GTOP; Messenger; CMAES, MIDACO; Parallelization

1 Introduction

Interplanetary trajectory optimization is a long standing challenge for space engineers and applied mathematicians alike. Since 2005 the Advanced Concept Team (ACT) of the European Space Agency (ESA) makes publicly available a comprehensive benchmark database of global trajectory optimization problems (GTOP) corresponding to real-world missions like Cassini, Rosetta and Messenger. The Messenger (full mission) benchmark in the GTOP database is notably the most difficult instance among those set, resembling a fully accurate model of the original trajectory of the Messenger mission launched by NASA in 2004.

The GTOP database expresses each benchmark as optimization problem (1) with box-constraints, whereas the objective function $f(x)$ is considered as nonlinear black-box function depending on a n-dimensional real valued vector of decision variables x. The GTOP database addresses researchers to test and compare their optimization algorithms on the benchmark problems.

$$\begin{aligned} &\text{Minimize} \quad f(x) \qquad\qquad (x \in \mathbb{R}^n) \\ &\text{subject to:} \;\; x_l \leq \; x \; \leq x_u \quad (x_l, \; x_u \in \mathbb{R}^n) \end{aligned} \tag{1}$$

In [22] it was demonstrated that the MIDACO algorithm could solve many of the GTOP benchmarks to their putative best known solution within minutes to hours using its default parameters. However, in [22] it was also demonstrated, that the MIDACO algorithm failed to solve the hardest benchmark of this set, the Messenger (full mission) benchmark, to even a *near global* solution despite a massive run time of 24 hours.

This contribution now addresses exclusively the Messenger (full mission) benchmark and demonstrates an optimization approach to robustly solve this benchmark within 24 hours to its (putative) global optimal solution. The considered optimization approach is called MXHPC, which stands for *MIDACO Extension for High Performance Computing*. The MXHPC algorithm is a (massive) parallelization framework which executes and operates several instances of the MIDACO algorithm in parallel and has been especially developed for large-scale computer clusters.

This paper is structured as follows: The second section introduces the Messenger (full mission) benchmark and highlights its difficulty by presenting some preliminary numerical results obtained by CMAES [3] and MIDACO [20]. The third section describes the MXHPC algorithm in detail. The fourth section presents the numerical results obtained by MXHPC solving the Messenger (full mission) benchmark on a moderate computer cluster. Finally some conclusions are drawn.

2 The Messenger (full mission) benchmark

The Messenger (full mission) benchmark models an multi-gravity assist interplanetary space mission from Earth to Mercury, including three resonant flyby's at Mercury. The sequence of fly-by planets for this mission is given by Earth-Venus-Venus-Mercury-Mercury-Mercury-Mercury, whereas the first item is the start planet and the last item is the final target planet. The objective of this benchmark is to minimize the total ΔV (change in velocity) accumulated during the full mission, which can be interpreted as reducing the fuel consumption. The benchmark invokes 26 continuous decision variables which are described as follows (for details on hyperbolic trajectories, see Kemble [16]):

Table 1. Description of optimization variables for Messenger benchmark

Variable	Description
1	Launch day measured from 1-Jan 2000 (MJD2000)
2	Initial excess hyperbolic speed (km/sec)
3	Component of excess hyperbolic speed
4	Component of excess hyperbolic speed
$5 \sim 10$	Time interval between events (e.g. departure, fly-by, capture)
$11 \sim 16$	Fraction of the time interval after which DSM* occurs
$17 \sim 21$	Radius of flyby (in planet radii)
$22 \sim 26$	Angle measured in planet B plane of the planet approach vector

* DSM stands for *Deep Space Manoeuvre*

The Messenger (full mission) benchmark does not contain constraints, except lower and upper bounds on the 26 decision variables. Table 2 displays the best known solution (corresponding to an objective function value of f(x) = 1.95863 km/sec) together with the lower and upper bounds and their unit (if available). Note that the best known solution displayed in Table 2 corresponds closely to the data of the real Messenger trajectory and is believed to be globally optimal.

The Messenger (full mission) benchmark is part of the GTOP database which is a collection of (black-box) optimization problems resembling several real-world space mission trajectories. The instances of the GTOP database are known to be difficult to solve and have attracted a considerable amount of attention in the past. Many researchers have worked and published results on the GTOP database, for example [1], [2], [4], [5], [6], [9], [10], [12], [13], [15], [17], [18], [24] or [26]. A special feature of the GTOP database is that the actual global optimal solutions are in fact unknown and thus the ESA/ACT accepts and publishes any new solution that is at least 0.1% better (relative to the objective function value) than the current best known solution. Table 3 lists the individual GTOP benchmark instances (without the *Tandem* series) together with their number of solution submissions and the total time span between the first and last submission, measured in years.

From Table 3 it can be seen that it took the community in most cases several months to about a year to obtain the putative global optimal solution. From Table 3 it can also be seen that the Messenger (full mission) benchmark is an exception in this regard and stands out by the number of submitted solutions and the time span between its first and last submission. **Over 5 years** were required by the community to achieve the current best known solution to the Messenger (full mission) benchmark. This is a remarkable amount of time and reflects well the difficulty of this benchmark, about which the ESA states on their website [8]:

Table 2. Best known solution for Messenger (full mission)

Variable	Lower bound	Solution value	Upper bound	Unit
1	1900	2037.8595972244	2300	MJD2000
2	2.5	4.0500001697	4.05	km/sec
3	0	0.5567269199	1	n/a
4	0	0.6347532625	1	n/a
5	100	451.6575153013	500	days
6	100	224.6939374104	500	days
7	100	221.4390510408	500	days
8	100	266.0693628875	500	days
9	100	357.9584322778	500	days
10	100	534.1038782374	600	days
11	0.01	0.6378086222	0.99	days
12	0.01	0.7293472066	0.99	n/a
13	0.01	0.6981836705	0.99	n/a
14	0.01	0.7407197230	0.99	n/a
15	0.01	0.8289833176	0.99	n/a
16	0.01	0.9028496299	0.99	n/a
17	1.1	1.8337484775	6	n/a
18	1.1	1.1000000238	6	n/a
19	1.05	1.0499999523	6	n/a
20	1.05	1.0499999523	6	n/a
21	1.05	1.0499999523	6	n/a
22	$-\pi$	2.7481808788	π	n/a
23	$-\pi$	1.5952416573	π	n/a
24	$-\pi$	2.6241779073	π	n/a
25	$-\pi$	1.6276418577	π	n/a
26	$-\pi$	1.6058416537	π	n/a

Table 3. GTOP database benchmark problems

GTOP Benchmark Name	Number of submissions	Time between first and last submission
Cassini1	3	0.5 years
GTOC1	2	1.1 years
Messenger (reduced mission)	3	0.9 years
Messenger (full mission)	**10**	**5.7 years**
Cassini2	7	1.2 years
Rosetta	7	0.5 years
Sagas 2	1	*(only one submission)*

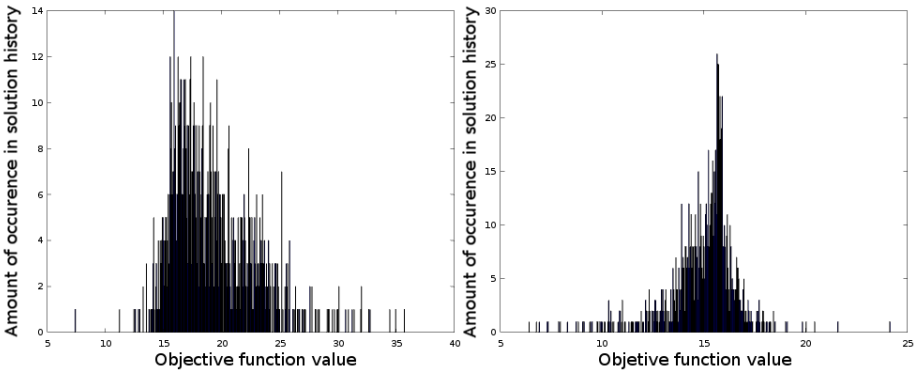
"it was hardly believable that a computer, given the fly-by sequence and an ample launch window, could design a good trajectory in complete autonomy without making use of additional problem knowledge."

ESA/ACT-GTOP website, 2016

2.1 Preliminary Numerical Results by CMAES and MIDACO

This sub-section further demonstrates the difficulty of the Messenger (full mission) benchmark by illustrating some preliminary numerical results achieved by the well-known CMAES [3] algorithm and the MIDACO [20] algorithm. Figure 1 presents a histogram of the solution objective function values obtained by 1000 independent test runs, performed once with CMAES and once with MIDACO.

Fig. 1. Histogram of CMAES (left side) and MIDACO (right side) solutions



The left side of Figure 1 displays the histogram of 1000 solution objective function values obtained by CMAES. The X-axis represents the objective function value and the Y-axis represents the frequency of such values among all those solution. For the numerical tests, the original CMAES implementation from Hansen [11] was used, all parameters set to default. Table 4 lists detailed information on the obtained results by CMAES from all test runs.

Table 4. CMAES results from 1000 test runs

Average solution $f(x)$	19.213 (ΔV)
Average function evaluation	232780
Average cpu-time	30.05 sec
Overall best solution $f(x)$	7.379 (ΔV)
Overall execution time	8.3 hours

In each performed test run the CMAES algorithm stopped by itself, if its internal standard deviation falls under a specific value (default 10^{-16} was used). On average the number of function evaluation performed by CMAES was 232780. Based on this number 1000 test runs were performed with MIDACO (all parameters set to default), using a fixed maximal number of 232780 function evaluation

in each such run. Table 5 lists detailed information on those test runs performed by MIDACO.

Table 5. MIDACO results from 1000 test runs

Average solution $f(x)$	14.961 (ΔV)
Average function evaluation	232780
Average cpu-time	29.01 sec
Overall best solution $f(x)$	6.399 (ΔV)
Overall execution time	8.1 hours

Comparing the results from the CMAES and MIDACO test runs, it can be seen that on average the CMAES algorithm stopped at a solution with an objective function value of 19.213, while the MIDACO algorithm stopped at a solution with an objective function value of 14.961. Both algorithms required a similar amount of time, which was about 30 second per run and totalled about 8 hours for all 1000 test run executions. The overall best solution reported by CMAES corresponded to an objective function value of $f(x) = \mathbf{7.379}$. The overall best solution reported by MIDACO corresponded to an objective function value of $f(x) = \mathbf{6.399}$.

As the best known solution to Messenger (full mission) benchmark has an objective function value of $f(x) = \mathbf{1.959}$, both algorithm (CMAES and MIDACO) have still been far away from reaching the global optimal solution in above setup, despite a significant time budget of 8 hours. This result does not come as a surprise, but further demonstrates the remarkable difficulty of this specific GTOP instance.

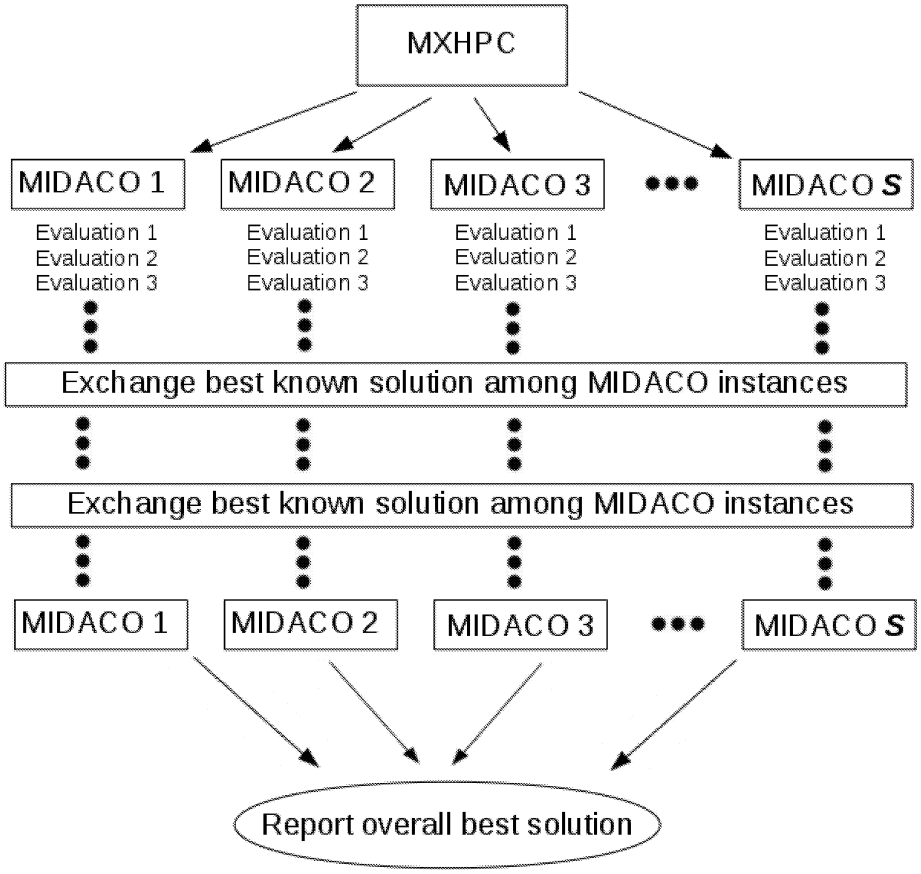
3 Description of the MXHPC Algorithm

The here considered algorithm represents a parallel framework for the MIDACO algorithm, which is an evolutionary black-box MINLP solver (see [20]). As this framework is particular suited for massive parallelization used in *High Performance Computing* (HPC) it is called MXHPC, which stands for *MIDACO Extension for HPC*. The basic purpose of the MXHPC algorithm is to execute several instances of MIDACO in parallel and manage the exchange of best known solution among those MIDACO instances.

Figure 2 illustrates how the MXHPC algorithm executes a number of $\boldsymbol{S}$ different instances of MIDACO in parallel. In regard to the well known Master/Slave concept in distributed computing, the indivudal MIDACO instances can be referred to as slaves, while the MXHPC algorithm can be referred to as master. In evolutionary algorithms such approach is also denoted as coarse-grained parallelization. Note in Figure 2 that the best known solution is exchanged by

MXHPC between individual MIDACO instances at a certain frequency (measured in function evaluation).

Fig. 2. Illustration of the MXHPC, executing S instances of MIDACO in parallel.



The MXHPC algorithm implies several individual parameters, this is the number of MIDACO instances, the exchange frequency of current best known solution and the survival rate of individual MIDACO instances at exchange times:

The considered exchange mechanism of best known solutions among individual MIDACO instances should be explained in more detail now, as this algorithmic step resembles the most sensitive part of the MXHPC algorithm. Let ***survive*** be the percentage (e.g. 25%) of surviving MIDACO instances at some

Parameter	Description
$\boldsymbol{S}$	Number of MIDACO instances (also called *slaves*)
exchange	Solution exchange frequency among slaves
survive	Survival rate (in percentage) among slaves

exchange (e.g. 1,000,000 function evaluation) time of the MXHPC algorithm. Then, at an exchange time, MXHPC will first collect the current best solutions of each of the $\boldsymbol{S}$ individual MIDACO instances and identifies the ***survive*** (e.g. 25%) best among them. Those MIDACO instances, which hold one of those best solutions, will be unchanged (thus the instance "survives" the exchange procedure). All other MIDACO will be restarted using the overall best known solution as starting point.

Readers with a deeper interest in the algorithmic details of MIDACO are referred to Schlueter et al. [19].

4 Numerical Results of MXHPC on the Messenger (full mission) benchmark

This section presents the numerical results obtained by MXHPC on the Messenger (full mission) benchmark. Like in [22] the optimization process was split into two different stages, one basic run (from scratch) and one refinement run. For both stages a time budget of 12 hours was considered. In total, ten tests (each consisting of a basic and a refinement run) have been conducted on Messenger (full mission). All tests have been conducted on the same Fujitsu FX10 cluster consisting of 64 Sparc64 cpu chips. The results of the basic runs are given in Subsection 4.1. The results of the refinement runs are given in Subsection 4.2.

4.1 Basic Runs on Messenger (full mission)

This subsection presents and discusses ten individual test runs of MXHPC on the Messenger (full mission) benchmark from scratch (using a random starting point). The here considered parameter specifications of the MXHPC algorithm (see Section 3) are as follows:

Parameter	Description
$\boldsymbol{S}$	1000
exchange	1,000,000
survive	25%

Table 6 reports the objective function value (ΔV), the number of MIDACO restarts within MXHPC and the number of total function evaluation for each of the ten individual basic runs of MXHPC on Messenger (full mission). Each

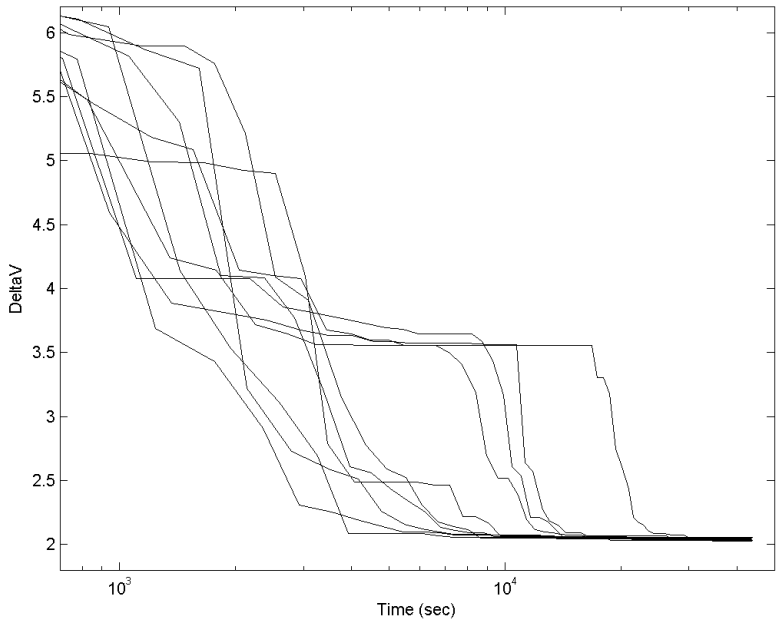
test run setup differs only in the specific seed for the pseudo random number generator used within MXHPC. Furthermore, Table 6 displays the time, when MXHPC (b)reached the objective function value of 2.113, which corresponds to the overall best reported solution reported by a group (Stracquadanio et al. [24]) not associated with MIDACO. Note that given the difficulty of Messenger (full mission), an objective function value of 2.113 can already be considered as remarkable good, whereas such a solution is still around 8% above the overall best known one of 1.959.

Table 6. 10 Test runs of MXHPC on MessFull

Seed	$f(x)$	*Restarts*	*Total Eval*	(B)*reach* $f(x) = 2.113$
1	2.0225	53,250	$70{,}000 \times 10^6$	9430 ($\sim$ 2.6 hours)
2	2.0295	52,500	$69{,}000 \times 10^6$	7261 ($\sim$ 2.1 hours)
3	2.0313	54,750	$72{,}000 \times 10^6$	13284 ($\sim$ 3.7 hours)
4	2.0481	54,000	$71{,}000 \times 10^6$	12344 ($\sim$ 3.4 hours)
5	2.0449	51,000	$67{,}000 \times 10^6$	5170 ($\sim$ 1.4 hours)
6	2.0481	36,000	$47{,}000 \times 10^6$	14104 ($\sim$ 3.9 hours)
7	2.0379	51,000	$67{,}000 \times 10^6$	3563 ($\sim$ 1.0 hours)
8	2.0441	52,500	$69{,}000 \times 10^6$	7794 ($\sim$ 2.2 hours)
9	2.0528	54,001	$71{,}000 \times 10^6$	23273 ($\sim$ 6.5 hours)
10	2.0263	51,750	$68{,}000 \times 10^6$	6412 ($\sim$ 1.8 hours)

From Table 6 it can be seen that in every test run a solution close to $f(x) = 2.0$ could be achieved, while the mark of $\Delta V = 2.113$ was (b)reached within one to six hours. It can be further seen that the number of total function evaluation necessary to achieve such objective function values ranges around 7×10^{10} (in words: seventy thousand million), which reflects well the complexity of this optimization problem. The number of MIDACO restarts within MXHPC ranged around fifty thousands. In addition to the results in Table 6, Figure 3 illustrates the convergence curves of all ten runs in regard to the cpu time measured in seconds.

Fig. 3. Convergence curves of ten basic runs (semi-log scale)



In conclusion on Table 6 and Figure 3 it can be stated that MXHPC is able to robustly solve Messenger (full mission) from scratch to an assumed near global solution, which is significantly better than the best result yet published by a different approach (see Stracquadanio et al. [24]).

4.2 Refinement Runs on Messenger (full mission)

This subsection presents the results of the refinement runs for each solution obtained by the previous basic run illustrated in Subsection 4.1. Particular this means that each MXHPC solution from Table 6 (with an objective function value close above 2.0) was used as starting point for each individual refinement run presented in this subsection. The MXHPC algorithm setup is identical to the on used for the basic run, except one MIDACO parameter. This parameter is the *FOCUS* parameter (see [23]) which is used to concentrate the search process of each individual MIDACO instance within MXHPC around the submitted starting point (called $X0$). While in Subsection 4.1 the default value of zero was assumed for FOCUS, here a value of 10,000[1] is used for all test runs. Note that

[1] Note that this is the same value for the FOCUS parameter as used for refinement runs in [22]

a FOCUS value of 10,000 will imply a shrinking of the standard devaiation in MIDACO's ACO multi-kernel gauss probability functions (see Schlueter et al. [19]) by a factor of 10,000 times, which is a massive factor that further illustrates the sensitivity of the Messenger benchmark.

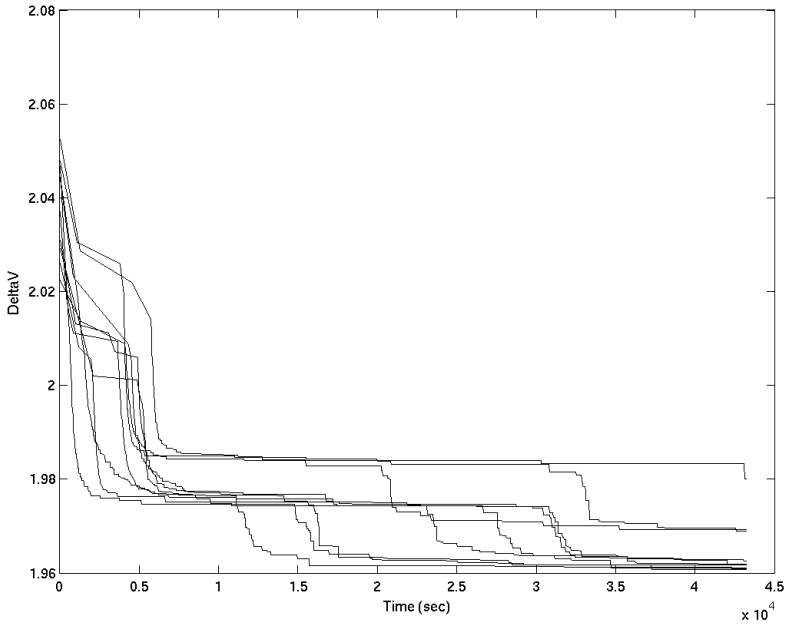
Table 7 shows the obtained result in regard to the final objective function value (ΔV) and its percentage (calculated as $100 \times (f(x) - 1.959)/1.959$) distance to current best known solution of 1.959.

Table 7. Refinement runs for solutions obtained in Section 4.1

X0 from Seed	$f(x)$	Distance to Best Known
1	1.9610	0.1 %
2	1.9609	0.1 %
3	1.9617	0.1 %
4	1.9607	0.1 %
5	1.9626	0.2 %
6	1.9801	1.1 %
7	1.9619	0.1 %
8	1.9689	0.5 %
9	1.9693	0.5 %
10	1.9619	0.2 %

From Table 7 it can be seen that in 5 out of 10 cases a final objective value close as 0.1% to the current best known one is obtained. In the worst case ($X0$ from seed 6), a solution with an objective close as 1.1% was obtained. In addition to the results in Table 7, Figure 4 illustrates the convergence curves of all ten refinement runs. Note in Figure 4 that the worst case run (corresponding to a final objective function value of about 1.98) appears as rather isolated result.

Fig. 4. Convergence curves of ten refinement runs (semi-log scale)



5 Conclusions

This contribution presented a rigorous study on the numerical optimization of ESA's Messenger (full mission) benchmark which is an exceptional difficult problem to solve. A novel algorithm, called MXHPC, was introduced. This algorithm acts as meta-algorithm on a computer cluster to operate in parallel several instances of the MIDACO algorithm. The MIDACO algorithm has previously shown to be efficient on interplanetary space mission design (see e.g. [21] or [22]). The numerical MXHPC results presented here demonstrate that it is possible to solve the Messenger (full mission) benchmark close to its putative global optimal solution within a single day. The results presented in section 4 show that with a robustness of 5 out of 10 runs, a solution as close as 0.1% to the best known one was obtained within 24 hours of run time on a 64 node Fujitsu FX10 cluster. In the worst case, a still remarkable good solution of just 1.1% above the best known one was obtained. These results further fortify the effectiveness of evolutionary computing for highly complex real-world applications that were previously considered intractable.

Future research might focus on both, improving the algorithmic performance of MXHPC and MIDACO as well as performing numerical test on larger computer clusters, having the ultimate goal to further reduce the overall required time to solve the Messenger (full mission) benchmark to few hours or even minutes.

Acknowledgement

The authors are grateful to the Advanced Concept Team (ACT) of the European Space Agency (ESA) and particular Dario Izzo for providing and maintaining the GTOP database. The first author would further like to thank the European Space Agency (ESA-ESTEC/Contract No. 21943/08/NL/ST) and EADS Astrium Ltd (Stevenage, UK) for their support on the MIDACO development.

References

1. Addis, B., Cassioli, A., Locatelli, M. & Schoen, F., Global optimization for the design of space trajectories. Comput. Optim. Appl. 48(3), 635-652, 2011.
2. Ampatzis, C., & Izzo, D.: Machine learning techniques for approximation of objective functions in trajectory optimisation, Proc. Int. Conf. Artificial Intelligence in Space (IJCAI), 2009.
3. Auger A., Hansen N.: A Restart CMA Evolution Strategy With Increasing Population Size. IEEE Congress on Evolutionary Computation, Proceedings. IEEE. pp. 17691776 (2005)
4. Biazzini, M., Banhelyi, B., Montresor, A. & Jelasity, M: Distributed Hyper-heuristics for Real Parameter Optimization, Proc. 11th Ann. Conf. Genetic and Evolutionary Computation (GECCO), 1339-1346, 2009.
5. Biscani, F., Izzo, D. & Yam, C.H.: A Global Optimisation Toolbox for Massively Parallel Engineering Optimisation, Proc. 4th Int. Conf. Astrodynamics Tools and Techniques (ICATT), 2010.
6. Danoy, G., Pinto, F.,G., Dorronsoro, B. & Bouvry, P.: New State-Of-The-Art Results For Cassini2 Global Trajectory Optimization Problem, Acta Futura 5, 65-72, 2012.
7. European Space Agency (ESA) and Advanced Concepts Team (ACT). GTOP database - global optimisation trajectory problems and solutions, Software available at `http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html`, 2016.
8. European Space Agency (ESA) and Advanced Concepts Team (ACT). GTOP database: Messenger (Full Mission) Instance , Software available at `http://www.esa.int/gsp/ACT/inf/projects/gtop/messenger_full.html`, 2016.
9. Gad, A.,H.,G.,E.: Space trajectories optimization using variable-chromosome-length genetic algorithms, PhD-Thesis, Michigan Technological University, USA, 2011.
10. Gruber, A.: Multi Gravity Assist Optimierung mittels Evolutionsstrategien, BSc-Thesis, Vienna University of Technology, Austria, 2009.
11. Hansen N.: The CMA Evolution Strategy, Sofware available at `https://www.lri.fr/~hansen/cmaesintro.html`, 2016.
12. Henderson, T.,A.: A Learning Approach To Sampling Optimization: Applications in Astrodynamics, PhD-Theis, Texas A & M University, USA, 2013.

13. Islam, S.K.M., Roy, S.G.S. & Suganthan, P.N.: An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Transactions on Systems, Man and Cybernetics 42(2), 482-500, 2012
14. Izzo, D.: 1st ACT global trajectory optimisation competition: Problem description and summary of the results Acta Astronaut. 61(9), 731-734, 2007.
15. Izzo, D.: Global Optimization and Space Pruning for Spacecraft Trajectory Design. *Spacecraft Trajectory Optimization* Conway, B. (Eds.), Cambridge University Press, 178-199, 2010.
16. Kemble, S.: Interplanetary Mission Analysis and Design. *Astronautical Engineering*, Springer , 2006.
17. Lancinskas, A., Zilinskas, J. & Ortigosa, P., M.: Investigation of parallel particle swarm optimization algorithm with reduction of the search area, Proc. Int. Conf. Cluster Computing Workshops and Posters (IEEE), 2010.
18. Musegaas, P.: Optimization of Space Trajectories Including Multiple Gravity Assists and Deep Space Maneuvers, *MSc Thesis*, Delft University of Technology, Netherlands, 2012.
19. Schlueter, M., Egea, J.A. & Banga, J.R.: Extended ant colony optimization for non-convex mixed integer nonlinear programming, Comput. Oper Res. 36(7), 2217-2229, 2009.
20. Schlueter, M., Gerdts, M. & Rueckmann, J.J.: A numerical study of MIDACO on 100 MINLP benchmarks, Optimization 61(7), 873-900, 2012.
21. Schlueter M., Erb S., Gerdts M., Kemble S., & Rueckmann J.J.: MIDACO on MINLP Space Applications. Advances in Space Research, 51(7), 1116-1131, 2013.
22. Schlueter M.: MIDACO Software Performance on Interplanetary Trajectory Benchmarks. Adv. Space Res., 54(4), pp. 744-754, 2014.
23. Schlueter M. & Munetomo M.: Introduction to MIDACO-SOLVER Software. Technical Report, HUSCAP, Hokkaido University, Japan, 2013c.
24. Stracquadanio, G., La Ferla, A., De Felice, M. & Nicosia, G.: Design of robust space trajectories, Proc. 31st Int. Conf. Artificial Intelligence (SGAI), 2011.
25. M. Ceriotti, M. Vasile: MGA trajectory planning with an ACO-inspired algorithm, Acta Astronautica, 67 (9-10). pp. 1202-1217, 2010.
26. Vinko, T. & Izzo, D.: Global Optimisation Heuristics and Test Problems for Preliminary Spacecraft Trajectory Design, European Space Agency, ACT Tec. Rept. ACT-TNT-MAD-GOHTPPSTD, 2008.