New state-of-the-art results on ESA's Messenger Space Mission Benchmark

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

*Information Initiative Center, Hokkaido University, Sapporo 060-0811, Japan.
†AIST-Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory
National Institute of Advanced Industrial Science and Technology Tokyo, Japan.
{schlueter@midaco-solver.com,mohamed.attia@aist.go.jp,munetomo@iic.hokudai.ac.jp}

Abstract. This contribution presents new state-of-the-art results for ESA's Messenger space mission benchmark, which is arguably one of the most difficult benchmarks available. The European Space Agency (ESA) created a continuous mid-scale black-box optimization benchmark which resembles an accurate model of NASA's 2004 launched Messenger interplanetary space probe trajectory. By applying an evolutionary optimization algorithm (MXHPC/ MIDACO) that relies on massive parallelization, it is demonstrated that it is possible to robustly solve this benchmark to a near global optimal solution within one hour on a computer cluster with 1000 CPU cores. This is a significant improvement over the previously in 2017 published state-of-the-art results where it was demonstrated for the first time, that the Messenger benchmark could be solved in a fully automatic way and where it took about 12 hours to achieve a near optimal solution. The here presented results fortify the effectiveness of massively parallelized evolutionary computing for complex real-world problems which have been previously considered intractable.

Keywords: Messenger Mission, Space Trajectory, Parallelization

1 Introduction

The optimization of interplanetary space trajectories is a long standing challenge for space engineers and applied mathematicians alike. The European Space Agency (ESA) created a publicly available comprehensive benchmark database of global trajectory optimization problems, known as 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 an accurate model of the entire trajectory of the original 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.

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

The benchmark instances of the GTOP database are known to be very 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], [3], [5], [6], [8], [10], [12], [13], [15], [17], [18], [19], [26] or [28]. 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 a new solution that is at least 0.1% better (relative to the objective function value) than the current best known solution. Table 1 lists the individual GTOP benchmark instances together with their number of solution submissions and the total time span between the first and last submission, measured in years. Note that as of 2020 the original GTOP database is no longer actively maintained by ESA. However an extended version, named GTOPX, continues the original GTOP source code base and introduces

Table 1. GTOP database benchmark problems

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

some minor improvements and simplified user-handling for the programming languages C/C++, Python and Matlab. The GTOPX software is freely available for downloaded at [11].

From Table 1 it can be seen that the Messenger (full mission) benchmark [9], [11] stands out as being by far the most difficult instance to solve. In most cases it took the community several months to about a year to obtain the putative global optimal solution, however the Messenger (full mission) benchmark is an exception in this regard and required a significant amount of submitted solutions and time span between its first and last submission. Well **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 the difficulty of this benchmark, about which the ESA stated on their website [9]:

"before the remarkable results...were found, ***it was hardly believable*** *that a computer...could design a good trajectory in complete autonomy without making use of additional problem knowledge."*

ESA/ACT-GTOP website, 2020 [9]

This contribution addresses exclusively the Messenger (full mission) benchmark and demonstrates that it is possible to robustly solve this instance close to its putative global optimal solution within one hour on the Hokkaido University HUCC Grand Chariot computer cluster [14], using 1000 cores for distributed computing. The considered optimization algorithm 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. The here presented results are a significant improvement over the previous state-of-the-art results published in 2017 [24], where it took 12 hours of computing time to achieve similar results on a cluster of comparable computing power.

This paper is structured as follows: The second section introduces the Messenger (full mission) benchmark and highlights its difficulty by referring to some previously published numerical results. 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 computer cluster. Finally some conclusions are drawn.

2 The Messenger (full mission) benchmark

The Messenger (full mission) benchmark [9], [11] 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.

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:

Table 2. Optimization variables for Messenger benchmark

Variable	Description
1	Launch day measured from 1-Jan 2000
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
$11 \sim 16$	Fraction of the time interval after DSM*
$17 \sim 21$	Radius of flyby (in planet radii)
$22 \sim 26$	Angle measured in planet B plane

* DSM stands for *Deep Space Manoeuvre*

The best known solution to the problem was obtained in 2014 by the MXHPC/MIDACO optimization software [22] and holds an objective function value of $f(x) = 1.959.$[1].

Table 3. 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

[1] Mingcheng Zuo (China Uni. of Geoscience) was able to refine this solution, so it rounds to an objective function value of $f(x) = 1.958$.

2.1 Published results on Messenger (full mission)

While being publicly available for over ten years now, published numerical results on the Messenger (full mission) benchmark remain very few only. This fact seems to stem from the tremendous difficulty to solve this problem instance. To our best knowledge, Table 4 lists all current available publications reporting numerical results on Messenger (full mission) in chronological order.

Table 4. Published results on Messenger (full mission)

Date	Author(s)	Ref	Algorithm	Best f(x)
2010	Biscani et. al.	[6]	PAGMO	3.950
2011	Stracquadanio et. al.	[26]	SADE	2.970
2011	Bryan	[7]	IGATO	7.648
2014	Schlueter	[23]	MIDACO	3.774
2017	Schlueter et al.	[24]	MXHPC	1.961
2019	Shuka	[25]	PASS	8.357

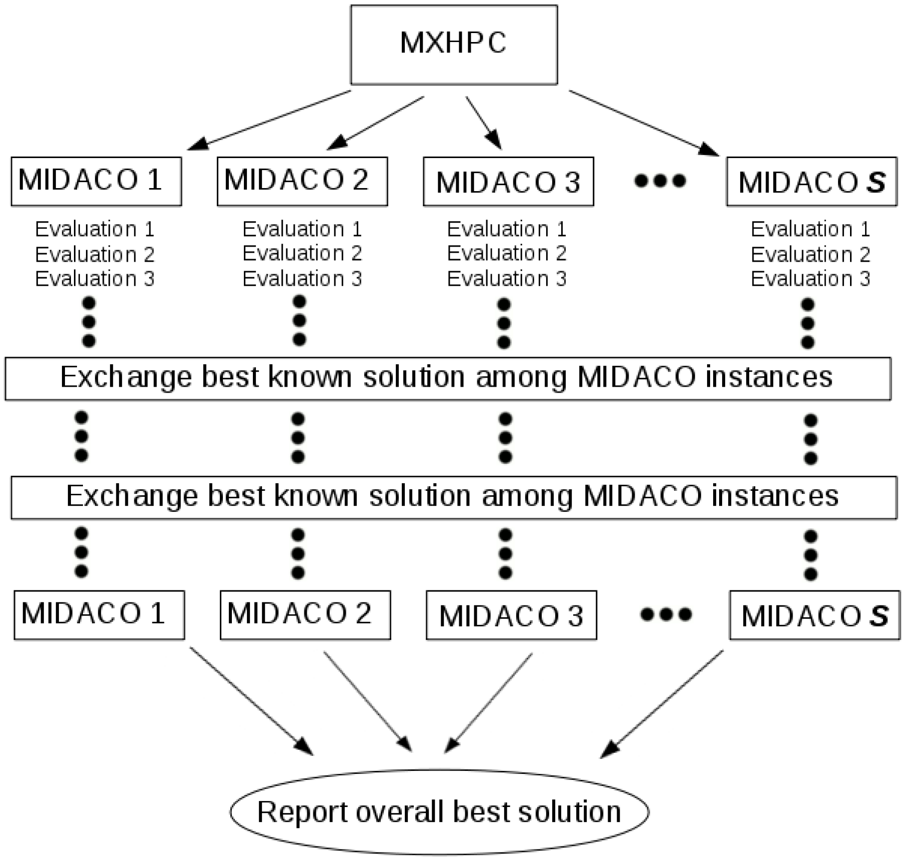
Table 4 lists the publication date, authors, reference, name of the considered algorithm together with the overall best objective function value f(x) obtained by that algorithm within that particular study. From Table 4 it can be seen that published results significantly vary and only the 2017 publication achieved a value close to the best known solution of 1.959. It is to note that the 2017 study (Schlueter et al. [24]) and 2019 study (Shunka [25]) both applied massive parallelization to execute their algorithms. The drastic difference in the best achieved solutions between those two studies (1.961 vs 8.357) illustrates that the use of a super-computer alone is not sufficient to solve the Messenger (full mission) benchmark and that instead the algorithmic element is crucial.

3 The MXHPC/MIDACO Algorithm

The here considered algorithm is called MXHPC and was originally introduced in 2017, see [24]. MXHPC is a parallelization framework built on top of MIDACO, which is an evolutionary black-box solver, see [21]. 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 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. The here presented version of MXHPC differs from the original proposed one by a dynamic instead of a static exchange rule, which is illustrated in Section 3.1.

Figure 1 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 individual 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 1 that the best known solution is exchanged by MXHPC between individual MIDACO instances at a certain frequency (measured in function evaluation).

Fig. 1. Illustration of the MXHPC, executing $\boldsymbol{S}$ 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:

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

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 ***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 [20].

3.1 New Modification: Dynamic Exchange

In contrast to the static exchange rule used within MXHPC in the previous publication from 2017 [24], a dynamic exchange rule is considered in this study. Based on some initial value (called *basevalue*), the evaluation budget of each individual MIDACO run within the MXHPC framework (see Figure 1) is linear increased successively. The pseudo code in Algorithm 1 describes in detail how the evaluation budget of each individual MIDACO instance is calculated, according to the successive number of exchanges. Note that a base value of 100000 was used for the numerical tests presented in Section 4.

Algorithm 1: Dynamic Exchange (pseudo code)

```
set basevalue = 100000
initialize evaluationbudget = 0

for exchange = 1 : ∞
    evaluationbudget = evaluationbudget +
                       exchange × basevalue
end
```

This dynamic exchange rule is based on the idea that with further progress each individual MIDACO instance within MXHPC requires more time (aka more function evaluation) to achieve progress, while such large budgets are not as useful in the beginning of the MXHPC execution. According to the pseudo code in Algorithm 1, the individual evaluation budgets of MIDACO will look as following:

eval-budget for MIDACO until 1st exchange: 100000
eval-budget for MIDACO until 2nd exchange: 300000
eval-budget for MIDACO until 3rd exchange: 600000
and so on ...

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. All results were calculated on the Hokudai supercomputer (HUCC Grand Chariot [14]) utilizing 1000 cores for distributed computing, which are composed of Intel Xeon Gold 6148 CPU's with a clock rate of 2.7 GHz. Ten independent test runs of MXHPC have been applied, each using the original lower bounds (see Table 3) as starting point and a different random seed for MIDACO's internal pseudo random number generator. Each individual test run was allowed to execute for one hour and then stopped automatically. The following parameters have been used for the MXHPC algorithm:

Parameter	Description
S	1000
exchange *(basevalue)*	100,000
survive	25%

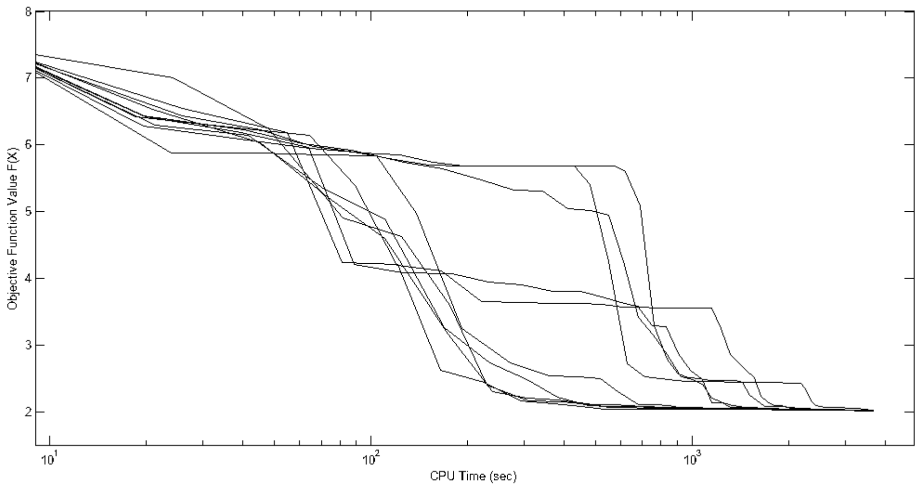
Table 5 reports the characteristics of each individual test run and the averaged values out of ten runs. The number of total function evaluation is displayed as multitude of 1000 in Table 5, corresponding to the 1000 cores which were used by MXHPC for the parallelization framework.

Table 5. 10 Test runs of MXHPC on Messenger

Run	f(x)	Evaluation	Time [sec]
1	2.0208	41,500,000 ×1000	3626.35
2	2.0245	41,500,000 ×1000	3637.20
3	2.0142	43,500,000 ×1000	3654.09
4	2.0242	48,500,000 ×1000	3639.93
5	2.0187	43,500,000 ×1000	3654.51
6	2.0182	45,500,000 ×1000	3602.26
7	2.0225	44,500,000 ×1000	3642.45
8	2.0264	47,500,000 ×1000	3667.15
9	2.0143	43,500,000 ×1000	3679.37
10	2.0220	42,500,000 ×1000	3647.33
Average:	2.0206	44,200,001 × 1000	3645.06

From Table 5 it can be seen that each run converged to an objective function value roughly above f(x) = 2.0 within one hour of run time. The averaged converged objective function value is f(x) = 2.0206 corresponding to the enormous number of about 44×10^9 function evaluation in total. In addition to Table 5, Figure 2 illustrates the convergence curves in semi-log scale of all ten test runs. Note in Figure 2 that all test runs have converged below an objective function value of f(x) = 6.0 within 1000 seconds (∼15 minutes) of CPU run time.

Fig. 2. Convergence curves of 10 individual MXHPC test runs



4.1 Comparison with previous results from 2017

This subsection gives an in-depth comparison between the newly presented results and the previously published ones in 2017, see [24]. Table 6 lists the ten from scratch[1] test runs of MXHPC with their corresponding final objective function value f(x) and corresponding number of total function evaluation. Both results, those from 2017 and 2020 , have been calculated on a cluster with 1000 cores. Back in 2017 this was a Fujitsu FX10 cluster [2] while now in 2020 this was the HUCC Grand Chariot cluster of the Hokkaido University [14].

[1] From scratch means here that the lower bounds have been used for all test runs. Those test runs therefore aim at exploring the entire search space and are not refinements of previous found solutions.

Table 6. Comparison regarding overall solution quality

	New Results (2020)		Previous Results (2017)	
Run	f(x)	Evaluation	f(x)	Evaluation
1	2.0208	41,500 $\times 10^6$	2.0225	70,000 $\times 10^6$
2	2.0245	41,500 $\times 10^6$	2.0295	69,000 $\times 10^6$
3	2.0142	43,500 $\times 10^6$	2.0313	72,000 $\times 10^6$
4	2.0242	48,500 $\times 10^6$	2.0481	71,000 $\times 10^6$
5	2.0187	43,500 $\times 10^6$	2.0449	67,000 $\times 10^6$
6	2.0182	45,500 $\times 10^6$	2.0481	47,000 $\times 10^6$
7	2.0225	44,500 $\times 10^6$	2.0379	67,000 $\times 10^6$
8	2.0264	47,500 $\times 10^6$	2.0441	69,000 $\times 10^6$
9	2.0143	43,500 $\times 10^6$	2.0528	71,000 $\times 10^6$
10	2.0220	42,500 $\times 10^6$	2.0263	68,000 $\times 10^6$
∅	2.0206	44,200 $\times 10^6$	2.0386	67,100 $\times 10^6$

From Table 6 it can be seen that both series of results averaged at a similar final solution objective function value of 2.0206 and 2.0386. The percentual difference between those values is about 0.9%, which may appear small, but is in fact relevant in context of the difficulty of the Messenger benchmark. In regard to the number of total function valuation, the new results require roughly about half (58.85%) of the amount required in 2017. As the results from 2017 were calculated with a time limit of 12 hours for each run while the new results are required in only 1 hour for each run, the averaged number of function evaluation reveal that the HUCC Grand Chariot cluster performed around 5 times ($4.938 = 12\times(1\text{-}0.5885)$) faster than the Fujitsu FX10 cluster.

Given that the new results required only around half the amount of function evaluation as back in 2017 and that the averaged final solution objective function value still shows an improvement of 0.9% over the old results, it can be concluded that the dynamic exchange strategy (see Section 3.1) is improving the algorithmic performance about at least two times.

An interesting threshold value for the Messenger (full mission) benchmark is the objective function value of f(x) = 2.113. The solution corresponding to this value was obtained by G. Stracquadanio (Johns Hopkins University) and G. Nicosia (University of Catania) and was included as new record solution in the GTOP database on 10th April 2012. Such solution is an improvement over the previous published one in Stracquadanio et al. [26] and to this date it remains the best known solution that was found without utilizing the MIDACO algorithm. It is therefore the best known competitive result and can act as a reference to compare with. In [24] the CPU time to reach the threshold of 2.113 has been measured and reported. Table 7 reports for each of the ten current MXHPC test runs after what amount of function evaluation and CPU time and with what objective function value the threshold of 2.113 was breached. Additionally Table 7 included the CPU time to breach the threshold in the 2017 study [24].

Table 7. Comparison regarding breach of f(x) = 2.113

	New Results (2020)			Results (2017)
Run	f(x)	Evaluation	Time [sec]	Time [sec]
1	2.112	5,500 $\times 10^6$	467	9,430
2	2.091	5,500 $\times 10^6$	473	7,261
3	2.107	8,500 $\times 10^6$	678	13,284
4	2.084	25,500 $\times 10^6$	1,593	12,344
5	2.098	17,500 $\times 10^6$	1,331	5,170
6	2.068	18,500 $\times 10^6$	1,312	14,104
7	2.091	23,500 $\times 10^6$	1,801	3,563
8	2.090	33,500 $\times 10^6$	242	7,794
9	2.080	5,500 $\times 10^6$	450	23,273
10	2.101	5,500 $\times 10^6$	477	6,412
∅	2.092	14,900 $\times 10^6$	882.4	10,263.5

From Table 7 it can be seen that back in 2017 it took on average 10,263.5 seconds to breach the threshold value of 2.113, while it took 882.4 seconds on average in this study. That is a 11.6 times improvement in such regard. Note that in Table 7 the number of function evaluation required to breach the threshold is on average about 15×10^9, while the number of function evaluation performed in a full hour in Table 5 averages about 44×10^9. This means that about 1/3 of the total evaluation budget (aka CPU time) is spent by MXHPC on reaching a value of about 2.092 and then 2/3 of the budget is spent by MXHPC on further converging toward a value of about 2.0206. This observation indicates that the Messenger (full mission) benchmark is exceptionally difficult in both regards: Locating the global optimal valley and then converging into that area to the exact solution.

5 Conclusions

Since over ten years ESA's Messenger benchmark is publicly available and acts as one of world's most challenging real-world benchmarks, formulated as numerical black-box optimization problem. In 2017 it could be shown for the very first time, that it is possible to solve this benchmark in a fully automatic way by applying an evolutionary algorithm on a supercomputer. While few publications attempted to address the Messenger problem, none were able to solve it to a close optimal solution, except for the 2017 study (see Table 4). This is in particularly true for the sub-optimal results published in Shuka [25], which also applied various evolutionary strategies on a supercomputer.

The results presented in this contribution are a continuation of the 2017 study and exhibit a significant improvement in terms of CPU run-time and evaluation budget. While in 2017 it took about 12 hours, the here presented results require only one hour of run-time and about half the function evaluation budget to achieve a similar (even slightly better) solution quality (see Table 6). An in-depth analysis in Section 4.1 revealed that this performance gain was about five times due to the faster hardware and about two times due to the algorithmic change made within MXHPC described in Section 3.1.

While the novelty of the algorithmic contribution in this study is only incremental, the reported numerical results are still of significance to a broad community of researchers who utilize these kind of benchmarks. This is due to the tremendous difficulty of the Messenger benchmark, about which the ESA stated that it was hardly believable to be solvable in an automatic fashion at all (see quote in Section 1). While the 2017 study proved the automatic solubility of this benchmark, this study is able to reduce the required CPU run-time to a single hour to robustly achieve a near-optimal solution. Overall, the here presented results fortify the effectiveness of massively parallelized evolutionary computing for complex real-world problems which have been previously considered intractable.

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. AIST Artificial Intelligence Cloud (AAIC). https://www.airc.aist.go.jp/en/info_details/computer-resources.html (2020)
3. Ampatzis, C., & Izzo, D.: Machine learning techniques for approximation of objective functions in trajectory optimisation, Proc. Int. Conf. Artificial Intelligence in Space (IJCAI), 2009.
4. Auger A., Hansen N.: A Restart CMA Evolution Strategy With Increasing Population Size. IEEE Congress on Evolutionary Computation, Proceedings. IEEE. pp. 17691776 (2005)
5. 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.
6. 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.
7. Bryan J.M.: Global optimization of MGA-DSM problems using the Interplanetary Gravity Assist Trajectory Optimizer (IGATO), Master Thesis, California Polytechnic State University (USA), 2011
8. 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.
9. European Space Agency (ESA) and Advanced Concepts Team (ACT). GTOP database - global optimisation trajectory problems and solutions, archived webpage https://www.esa.int/gsp/ACT/projects/gtop/messenger_full/, 2020.
10. Gad, A.,H.,G.,E.: Space trajectories optimization using variable-chromosome-length genetic algorithms, PhD-Thesis, Michigan Technological University, USA, 2011.
11. GTOPX - Space Mission Benchmark Collection, software available at http://www.midaco-solver.com/index.php/about/benchmarks/gtopx, 2020.
12. Gruber, A.: Multi Gravity Assist Optimierung mittels Evolutionsstrategien, BSc-Thesis, Vienna University of Technology, Austria, 2009.
13. Henderson, T.,A.: A Learning Approach To Sampling Optimization: Applications in Astrodynamics, PhD-Theis, Texas A & M University, USA, 2013.
14. Hokaido University High-Performance Intercloud. https://www.hucc.hokudai.ac.jp/en/supercomputer/sc-overview/ (2020)
15. 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
16. Izzo, D.: 1st ACT global trajectory optimisation competition: Problem description and summary of the results Acta Astronaut. 61(9), 731-734, 2007.
17. Izzo, D.: Global Optimization and Space Pruning for Spacecraft Trajectory Design. *Spacecraft Trajectory Optimization* Conway, B. (Eds.), Cambridge University Press, 178-199, 2010.
18. 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.
19. Musegaas, P.: Optimization of Space Trajectories Including Multiple Gravity Assists and Deep Space Maneuvers, *MSc Thesis*, Delft University of Technology, Netherlands, 2012.
20. 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.
21. Schlueter, M., Gerdts, M. & Rueckmann, J.J.: A numerical study of MIDACO on 100 MINLP benchmarks, Optimization 61(7), 873-900, 2012a.
22. Schlueter M., Erb S., Gerdts M., Kemble S., & Rueckmann J.J.: MIDACO on MINLP Space Applications. Advances in Space Research, 51(7), 1116-1131, 2013a.
23. Schlueter M.: MIDACO Software Performance on Interplanetary Trajectory Benchmarks. Adv. Space Res., 54(4), pp. 744-754, 2014.
24. Schlueter M., Wahib M., & Munetomo M.: Numerical optimization of ESAs Messenger space mission benchmark Proceedings of the Evostar Conference. Proceedings of the Evostar Conference (Springer), Amsterdam, April 19-21, pp. 725-737, 2017.
25. Shuka R.: Parallele adaptive Schwarmsuche fuer Blackbox-Probleme, PhD-Thesis, Gottfried Wilhelm Leibniz University Hannover (Germany), 2018.
26. Stracquadanio, G., La Ferla, A., De Felice, M. & Nicosia, G.: Design of robust space trajectories, Proc. 31st Int. Conf. Artificial Intelligence (SGAI), 2011.
27. M. Ceriotti, M. Vasile: MGA trajectory planning with an ACO-inspired algorithm, Acta Astronautica, 67 (9-10). pp. 1202-1217, 2010.
28. 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.