Parallelization Impact on Many-Objective Optimization for Space Trajectory Design

Martin Schlueter, Chit Hong Yam, Takeshi Watanabe,Akira Oyama

***Abstract*—This contribution discusses a novel many-objective optimization algorithmthat combines anant colony optimization based decomposition approach with a massive parallelization framework. A rigorous numerical analysis on the impact of the two varying key factors of the here considered parallelization approach is presented. Those factors are the number of co-evaluated solution candidates within an individual ant colony algorithm and the number of individual ant colony algorithms itself.Aim of the presented method is to solve a many-objective application corresponding to the interplanetary space trajectory of the Cassini probe, launched by NASA in 1997. The provided numerical results indicate that comprehensive mission analysis via a many-objective approach is possible and that the presented approach is highly suitable for massive parallelization.**

***Index Terms*— Many-Objective Optimization, Ant Colony Optimization, Space Flight Trajectory, Parallelization**

I. Introduction

Numerical optimization is an essential requirement in science and technology. This contribution discusses optimization problems containing several objectives and which can be mathematically defined as follows:

$$Minimize \quad f_1(x), \dots, f_M(x) (x \in \mathbb{R}^n, n \in \mathbb{N})$$

$$subject\ to:$$

$$\begin{aligned} g_i(x) &= 0, i = 1, \dots, m_e \in \mathbb{N} \\ g_i(x) &\geq 0, i = m_e, \dots, m \in \mathbb{N} \end{aligned} \qquad (1)$$

$$and\ bound\ constraints:$$

$$x_l \leq x \leq x_u \ (x_l, x_u \in \mathbb{R}^n).$$

Without loss of generality, a constrained minimization problem of M objectives is described in (1). Note that in case of more than three objectives, the above multi-objective problem is alternatively referred toas many-objective problem. The investigation of many-objective optimization problems is rapidly developing and those problems have been recently studied for example in Aguirre et al. [1], Aguirre et al. [2], Duro et al. [3], Luecken et al. [4] or Yuang et al. [5].

Manuscript received July 23, 2015.This work was supported in part by the RIKEN Advanced Institute for Computational science (Proposal number hp140231).The co-author of this paper, Chit Hong Yam, is supported by the JSPS international postdoctoral research fellowship. Martin Schlueter, Chit Hong Yam, Takeshi Watanabe and Akira Oyama are with the Japanese Aerospace Exploration Agency (JAXA), Institute of Space and Astronautical Science (ISAS) 3-1-1 Yoshinodai, Chuo-ku, Sagamihara, Kanagawa 252-5210, JAPAN (e-mail: martin@flab.isas.jaxa.jp,chithongyam@gmail.com,watanabe@flab.isas.jaxa.jp, oyama@flab.isas.jaxa.jp).

Recently a new algorithm for multi- and many-objective optimization based on massively parallelized Ant Colony Optimization (ACO) was proposed in Schlueter et al. [6]. This algorithm is based on a continuous ACO, as described in detail in Schlueter et al. [7], rather than a combinatorial one, which has been showing particular promising performance on interplanetary space trajectory design (see [8]).

Keeping in mind that parallelization is a growing trend in computer architecture design, this algorithm is particular suitable to exploit (massive) parallelization. The parallelization of the algorithm happens in two regards: Firstly, several instances of ACO are executed in parallel and secondly, the function evaluation of each ACO instance is performed in parallel (also denoted as co-evaluation). Particular the parallel execution of function evaluation is a crucial criteria for the applicability of the algorithm to cpu-time intensive real-world applications.

The design of interplanetary space trajectories focuses traditionally on a 'primary' single objective, such as the ΔV (for chemical propulsion missions) or maximization of final spacecraft mass (for low-thrust missions [9]). Other objectives, such as time of flight and launch date, are normally treated as 'secondary' objectives and implicitly incorporated in the mission analysis in form of constraints. This means the mission designer needs to set some reasonable lower and upper bounds on those secondary objectives. As a consequence of such procedure, the solution space is restricted and might exclude potential interest solutions.

The approach followed in this contribution does not rely on constraints but on integrative mission design, treating the secondary objectives in a multi/many-objective way. The goal of this research is to avoid the potential pitfall of solution space shrinking and equip the mission designer with a more flexible tool. Note that multi-objective approaches towards the here considered Cassini mission (without deep space maneuvers) has also been studied Vasile and Zuiani[10] and Zuiani and Vasile [11].

While in Schlueter et al. [6] the validity of the proposed algorithm for space applications was investigated, this contribution presents a rigorous analysis of the impact of the two levels of parallelization in regard to the performance of the algorithm.

This paper is structured as follows: Section II describes the algorithmic approach to solve the many objective problem. Section III gives details on the considered space mission model. In Section IV achieved numerical results are presented and compared with another state-of-the-art many-objective algorithm. The paper finishes with some conclusions.

II. Optimization Approach

This Section describes the algorithm to solve the multi-objective optimization problem (1) described in the introduction. The here presented algorithm was recently introduced in Schlueter et al. [6]. It is based on a combination of a decomposition of the original multi-objective problem into a series of single-objective problem and a (massive) parallelization framework of individual ACO instances, each handling one of those single-objective problems. A detailed description of the ACO algorithm considered here can be found in Schlueter et al. [7].In regard to the constraints stated in problem in (1), for the sake of brevity, the set of all *x* that fulfill the feasibility constraints in (1) should be denoted as $\mathcal{F}$.

A. Decomposition based on Utopia-Nadir Balance

The decomposition approach considered here is based on so called utopia and nadir information. The utopiaU_i of an individual objective $f_i(x)$ represents the global minimum of the respective objective and is formally defined as follows:

$$U_i = \min\{f_i(x) \; \forall \; x \; \epsilon \; \mathcal{F}\}. \tag{2}$$

In contrast to the utopia U_i, the nadir N_i represents the worst objective function value for the respective $f_i(x)$among all solutions *x* which correspond to an utopia $U_k$of any other objective $f_k(x)$. The nadir $N_i$is formally defined as follows:

$$N_i = \max\{f_i(x) \; \forall \; x : \; \exists \; k \; \neq \; U_k\}. \tag{3}$$

Given the utopia and nadir information, a scalar function is introduced in the following which acts as an indicator for the *balance* of a solution*x*.Therefore the here presented decomposition approach is coined *Utopia-Nadir-Balance* decomposition. A graphical illustration of an equally weighted *Utopia-Nadir-Balance* for a two-dimensional objective function can be found in Fig. 1.

Now the *Utopia-Nadir-Balance*decomposition is described in detail. Let *S*be the integer value which denotes the amount of single-objective sub problems in which the original multi-objective problem is decomposed into. Let further *w*be a matrix of size O × S containing weights in *[0,1]* for each of the *S* sub problems.Given the utopia $U_i$and nadir $N_i$information is globally available among all sub problems, then a (weighted) distance $d_i^j(x)$ for a solution *x* in each objective $f_i(x)$according to a sub problem *j* is defined as follows:

$$d_i^j(x) = w_i^j \frac{f_i(x) - U_i}{N_i - U_i}. \tag{4}$$

Let the average distance $D_j(x)$ for a solution *x* according to a sub problem *j* should be defined as follows:

$$D_j(x) = \frac{\sum_{i=1}^{M} d_i^j(x)}{M}. \tag{5}$$

Given the distance d_i^j and the average distance $D_j(x)$, now the balance $B_j(x)$ of a solution *x* according to a sub problem *j* is defined as follows:

$$B_j(x) = \sum_{i=1}^{M} \left| d_i^j(x) - D_j(x) \right|, \tag{6}$$

where|∙| denotes some norm function, e.g. the absolute value. From equation (4), (5) and (6) it can be seen, that balance $B_j(x)$ expresses a measurement for the average distance of a solution *x* in regard to the utopia's and nadir's of each objective.Note that the value of $B_j(x)$ is bounded to the interval *[0,1]*.

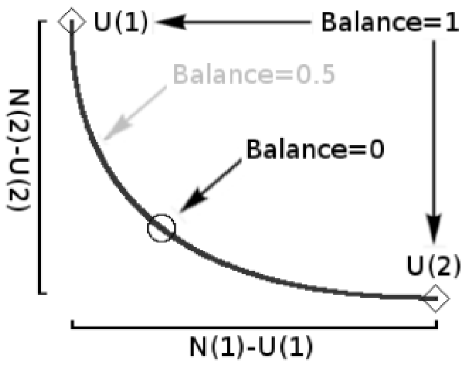

Fig. 1.Illustration of the balance concept.

Using the balance *B* , utopia *U* , nadir *N* and weights *w*, a single objective function *T* (denoted as *target* function)is defined for each *j*-thsubproblem:

$$T_j(x) = \sum_{i=1}^{M} w_i^j \frac{f_i(x) - U_i}{N_i - U_i} + B_j(x) \tag{7}$$

The set of single objective problems given by $T_1,...,T_M$therefore decomposes theoriginal multi-objective problem.

Note that in case of the very first initial execution ofthe ACO instances (illustrated in Fig. 3), no information about the current utopia *U* and nadir *N* is available.As a heuristic solution to this dilemma, the target function *T* is replaced for the veryfirst execution by a simple weighted sum over the objectives. For each *j*-thsubproblemthis function is then given as follows:

$$\hat{T}_j(x) = \sum_{i=1}^{M} w_i^j f_i(x). \tag{8}$$

Note that in Schlueter et al. [8] was in particular investigated, how the algorithmic performance depends on the target functions *T(x)* and its initial substitute given in equation (8).

B. Parallelization Framework

This subsection describes the parallelization framework, called *ACOMOD*, which executes and operates aset of *S* individual ACO algorithms in parallel. The *ACOMOD* framework classifies as a Master-Slave model, where each Slave represents an individual ACO algorithm, executed in an individual thread. Each of the *S* individual ACO algorithms is assigned to a different single objective problem, resulting from the decomposition of the original multi-objective problem described in Subsection II-A. While theoretically each ACO instance could operate in complete autonomy, the overall performance should be improved by exchanging utopia, nadir and best known solution information between individual ACO instances from time to time. At the very end, each individual ACO instance reports its set of non-dominated solutions found to the ACOMOD master framework, where this informationis processed to create the overall approximation of the pareto front. Fig. 3 illustrates the ACOMOD framework, executing several instance of ACO's with exchanging solutions and Fig. 2 illustrates the algorithm in a pseudo-code description.

```
**Algorithm 1** ACOMOD pseudo-code
Create $O \times S$ weights $w_i^j$
Create initial target functions $\hat{T}_j(x) = \sum_{i=1}^{M} w_i^j \cdot f_i(x)$
for all $j = 1, ..., S$ do
    Minimize $\hat{T}_j(x)$ by individual ACO instance
end for
for all Iteration =1,2,3,... do
    Update Utopia $U_i$ and Nadir $N_i$ for $i = 1, ..., O$
    Create Utopia-Nadir-Balance target functions $T_j(x)$
    for all $j = 1, ..., S$ do
        Minimize $T_j(x)$ by individual ACO instance
    end for
end for
Collect non-dominated solutions from ACO instances
```

Fig.2.Illustration of the ACOMOD pseudo-code

In order to be able to treat even very cpu-time intensive applications, the ACOMOD framework offers a second parallelization layer concerning the execution of objective and constraint function in each individual ACO instance. In regard to Fig. 3 this means that the individuals of each ACO instance can be co-evaluated in parallel.

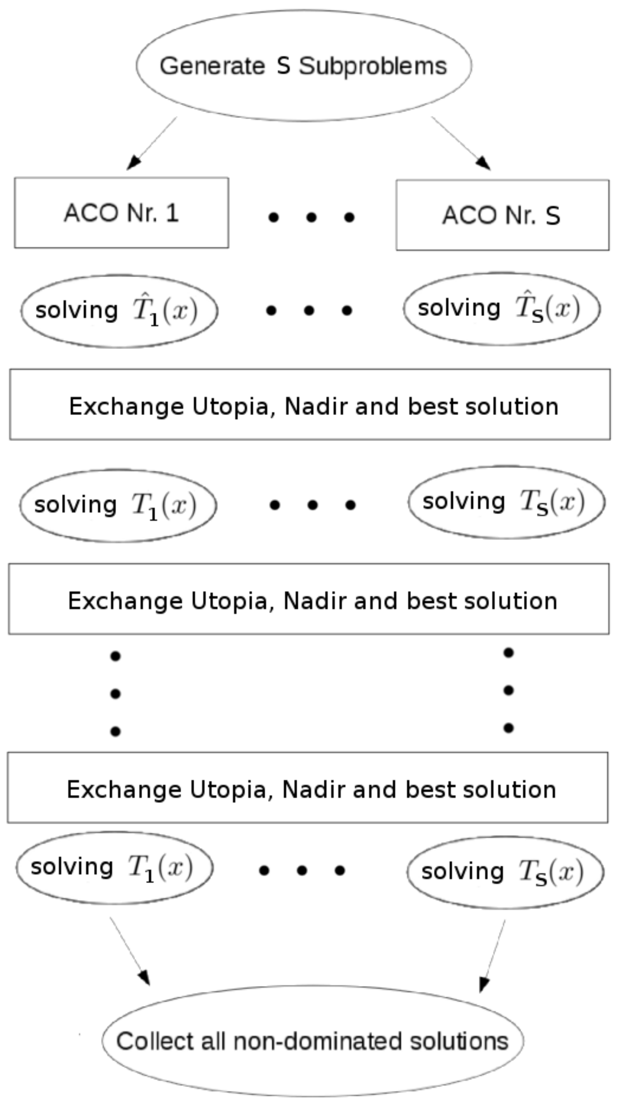
Generate S Subproblems
ACO Nr. 1
ACO Nr. S
solving $\hat{T}_1(x)$
solving $\hat{T}_S(x)$
Exchange Utopia, Nadir and best solution
solving $T_1(x)$
solving $T_S(x)$
Exchange Utopia, Nadir and best solution
Exchange Utopia, Nadir and best solution
solving $T_1(x)$
solving $T_S(x)$
Collect all non-dominated solutions

Fig.3.Illustration of the ACOMOD parallelization framework.

III. Many-Objective Space Trajectory Model

The here considered space mission is a simplified model of the Cassini mission to Saturn, launched by NASA in 1997 (see Fig. 4.)

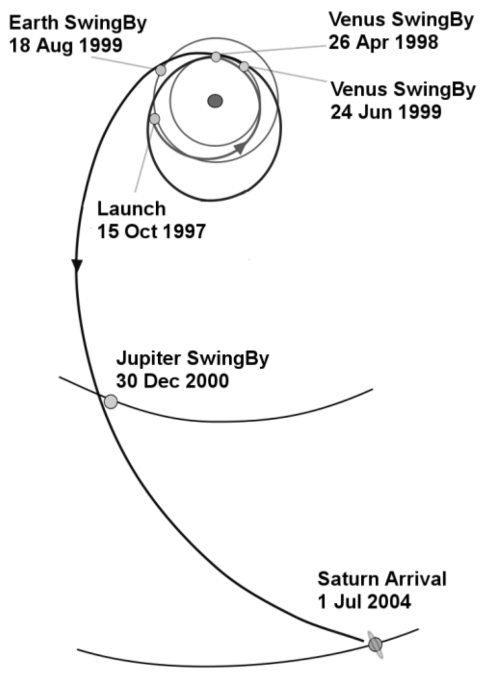

Fig.4.Illustration of the Cassini trajectory launched 1997.

The model is made available as interplanetary trajectory design problem in the European Space Agency (ESA) Global Trajectory Optimization Problems (GTOP) database[12]: The 'Cassini1' problem. In its original formulation by ESA, the model consists of 6 variables (see Table I with a single objective of the total ΔV including the launch V_∞at Earth, powered-flyby ΔVat each swing-by, and the arrival V_∞at Saturn. The trajectory is a two-body, patched-conic gravity assist model, as opposed to the more complicated MGA-DSM (with deep space maneuver) model.

TABLE I: Variables and Bounds

Var	Name	Lower & Upper Bound	Unit
x(1)	Launch Date	-1000 ~ 0	MJD2000
x(2)	TOF 1	30 ~ 400	Days
x(3)	TOF 2	100 ~ 470	Days
x(4)	TOF 3	30 ~ 400	Days
x(5)	TOF 4	400 ~ 2000	Days
x(6)	TOF 5	1000 ~ 6000	Days

In regard to the original single-objective formulation by ESA, here the model is extended to four objectives.

This is namely the ΔV, the launch V_∞, time of flight and the launch date (see Table II). While the ΔVor time of flight are commons objectives, the minimization of the departure time was considered here (namely *F3* in Table II), as this objective may occur in a mission projects where the target planet should be reached as soon as possible. For such a mission project a trajectory solution with exceptional short flight time but with a launch date far in the future would be inappropriate.

TABLE II: The Four Objectives

Objectives	Names	Units
F1	ΔV (excluding Launch Δ_∞)	Km/Sec
F2	Time of Flight	Days
F3	Launch Date	MJD2000
F4	LaunchΔ_∞	Km/Sec

IV. Computational Results

This section discusses numerical results achieved by the proposed optimization algorithm (see Section II) on the considered space mission benchmark (see Section III). While in Schlueter et al. [8] the numerical evaluation of validity of the *Utopia-Nadir-Balance* (see Section II-A) was investigated, here a rigorous investigating of the impact of parallelization is presented.

As described in Section II-B, the ACOMOD algorithm offers two options for parallelization, this is the number of individual ACO instances (see Fig. 3) and the number of co-evaluated individuals in each ACO. In order to investigate how the performance of ACOMOD depends on these two parallelization factors, several test runs with varying combinations of first level parallelization (thus the number of ACO instances) and second level parallelization (thus the number of co-evaluated individuals) are performed.

Two basic setups regarding the number of ACO instances within ACOMOD are considered, this is firstly 10 ACO instances and secondly 100 ACO instances. Given those two basic setups, the number of co-evaluated individuals is now varied from 1 over 10 and 100 up to 1,000.Given a fixed total function evaluation budget of 10,000,000 those settings imply the number of maximal generations per each ACO in each setup. For example, in case of ACOMOD with 10 ACO instances and a co-evaluation factor of 1 (which means no parallel evaluation), each ACO instance can perform 1,000,000 generations (and thus <u>many</u> sequential algorithmic steps). In case of ACOMOD with 100 ACO instances and a co-evaluation factor of 1,000, each ACO instance can perform only 100 generations (and thus <u>very few</u> sequential algorithmic steps).

TABLE III: Average HV Results with 10 ACO Instances

ACO Generations	Co-Evaluation	∅Hyper Volume
1,000,000	1	0.84205
100,000	10	0.85522
10,000	100	0.89968
1000	1,000	0.92951

TABLE IV: Average HV Results with 100 ACO Instances

ACO Generations	Co-Evaluation	∅Hyper Volume
100,000	1	0.90196
10,000	10	0.90905
1,000	100	0.94921
100	1,000	0.94979

Table III and Table IV display the average hyper volume (HV) results obtained for each individual setup executed with

10 test runs. It can be seen that the highest (where higher is better) average HV result of 0.94979 is achieved for the maximal parallelization with 100 ACO instances and a co-evaluation factor of 1,000. The lowest average HV result of 0.84205 was achieved with the lowest level of parallelization, which were 10 ACO instances with a co-evaluation factor of 1.

It is interesting to see from Table III and Table IV that the impact of the two parallelization factors is non-trivial.For example the average hyper volume result for the combination of 10 ACO instances with a co-evaluation factor of 1,000 yields better result than those combinations with 100 ACO instances with a co-evaluation factor of 1 or 10.In regard to the original motivation of this research, obtaining a highly parallelizable many-objective optimization algorithm, it is important to note that the overall best result from Table III and IV correspond to the highest parallelization of level one (number of ACO instances) and level two (co-evaluation factor).Note that the last setup with 100 ACO instances and a co-evaluation factor of 1,000 required the amount of 100,000 parallel threads, which is categorized as massive parallelization. All calculations in Table III and IV have been conducted on the K-Supercomputer.

Fig. 5 and Fig 6.illustrate the set of non-dominated solutions obtained by the best out of 10 individual test runs from the first setup in Table III (10 ACO instances with a co-evaluation factor of 1) and last setup of Table IV (100 ACO instances with a co-evaluation factor of 1,000) in regard to the first objective (*ΔV*) and second objective (time of flight). The set of non-dominated solution illustrated in Fig. 5 correspond to a hyper volume value of 0.87954. The set of non-dominated solution illustrated in Fig. 6 correspond to a hyper volume value of 0.95389.

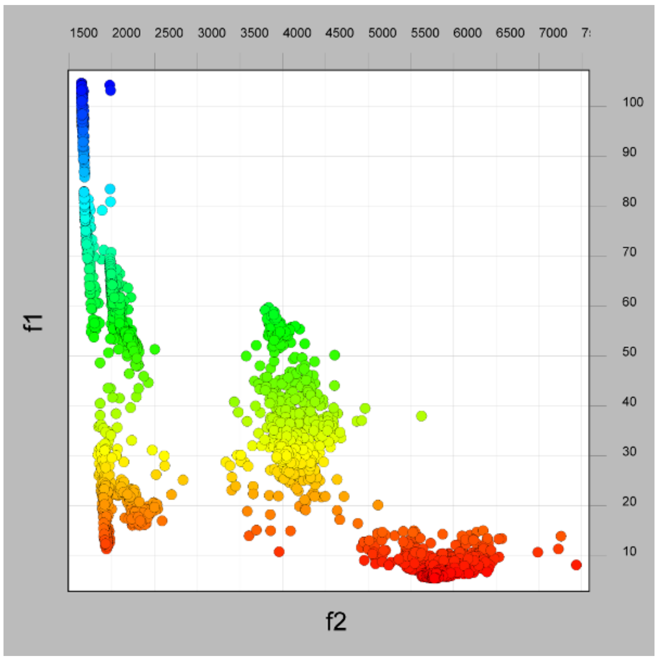

Fig.5.Non-dominated solutions from best run of the first setup (co-evaluation factor 1) of Table III.

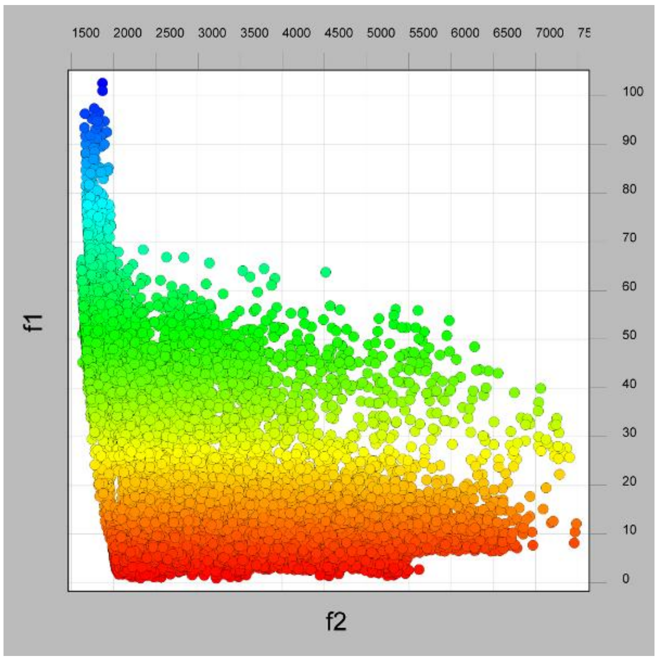

Fig.6.Non-dominated solutions from best run of the last setup (co-evaluation factor 1,000) in Table IV.

The illustration of the best individual run of the first setup in Table III was chosen, as this setup represents the lowest level of parallelization. The illustration of the best individual run of the last setup in Table III was chosen, as this setup represents the highest level of parallelization. Besides the significant difference in the value of the hyper volume between those two runs (HV=0.87954vs.HV= 0.95389), Fig. 5 and Fig. 6 illustrate well the different solution space coverage capacities of both setups, whereas a higher level of parallelization is clearly in favor here.

V. Conclusions

A new algorithm (called ACOMOD) for multi- and many-objective optimization was discussed and numerically evaluated on a space mission benchmark provided by the European Space Agency. The numerical analysis revealed that the proposed algorithm does highly benefit from (massive) parallelization and is therefore suitable for high performance computing facilities like the K-Supercomputer. The analysis further revealed that the impact of the two varying parallelization factors is non-trivial and needs further investigation. An important implication of the (massive) parallelization capability of the algorithm and in particular it’s second parallelization level (denoted as “co-evaluation”) is itsapplicability to even very cpu-time expensive real-world application.

Appendix

Here the individual hyper volume (HV) results of each individual numerical test run are displayed. The here displayed results correspond to the average values reported in Table III and Table IV in Section IV. Table V shows the HV results of the individual runs corresponding to 10 ACO instances and a co-evaluation factor of 1 and 10. Table VI shows the HV results of the individual runs corresponding to

10 ACO instances and a co-evaluation factor of 100 and 1,000. Table VII shows the HV results of the individual runs corresponding to 100 ACO instances and a co-evaluation factor of 1 and 10. Table VIII shows the HV results of the individual runs corresponding to 100 ACO instances and a co-evaluation factor of 100 and 1,000.

TABLE V: Individual HV results of numerical test runs with 10 ACO instances

Numerical runs with co-evaluation factor 1		Numerical runs with co-evaluation factor 10	
Run 1	0.6957159	Run 1	0.7735096
Run 2	0.8214306	Run 2	0.8725294
Run 3	0.8282290	Run 3	0.8458235
Run 4	0.8504792	Run 4	0.8479869
Run 5	0.8729078	Run 5	0.8777737
Run 6	0.8739988	Run 6	0.8436767
Run 7	0.8735564	Run 7	0.8751145
Run 8	0.8685478	Run 8	0.8708845
Run 9	0.8795437	Run 9	0.8664833
Run 10	0.8561217	Run 10	0.8784168

TABLE VI: Individual HV results of numerical test runs with 10 ACO instances

Numerical runs with co-evaluation factor 100		Numerical runs with co-evaluation factor 1,000	
Run 1	0.8944041	Run 1	0.9252429
Run 2	0.8855513	Run 2	0.9381606
Run 3	0.9068527	Run 3	0.9301337
Run 4	0.9143026	Run 4	0.9372895
Run 5	0.9050605	Run 5	0.9131399
Run 6	0.9110774	Run 6	0.9491826
Run 7	0.8992004	Run 7	0.9102161
Run 8	0.8992033	Run 8	0.9427224
Run 9	0.9011270	Run 9	0.9196478
Run 10	0.8800702	Run 10	0.9294061

TABLE VII: Individual HV results of numerical test runs with 100 ACO instances

Numerical runs with co-evaluation factor 1		Numerical runs with co-evaluation factor 10	
Run 1	0.8678095	Run 1	0.8505769
Run 2	0.8879897	Run 2	0.8964099
Run 3	0.8992584	Run 3	0.9023381
Run 4	0.9060852	Run 4	0.9163643
Run 5	0.9048196	Run 5	0.9194974
Run 6	0.9116159	Run 6	0.9218590
Run 7	0.9129544	Run 7	0.9194105
Run 8	0.9114331	Run 8	0.9227522
Run 9	0.9095349	Run 9	0.9254553
Run 10	0.9080850	Run 10	0.9158429

TABLE VIII: Individual HV results of numerical test runs with 100 ACO instances

Numerical runs with co-evaluation factor 100		Numerical runs with co-evaluation factor 1,000	
Run 1	0.9514614	Run 1	0.9536307
Run 2	0.9536226	Run 2	0.9538555
Run 3	0.9343257	Run 3	0.9498546
Run 4	0.9497940	Run 4	0.9538692
Run 5	0.9529386	Run 5	0.9342381
Run 6	0.9522419	Run 6	0.9525265
Run 7	0.9415437	Run 7	0.9498546
Run 8	0.9513782	Run 8	0.9542652
Run 9	0.9509938	Run 9	0.9419219
Run 10	0.9538493	Run 10	0.9538889

ACKNOWLEDGMENT

The numerical results have been obtained by using the K computerat the RIKEN Advanced Institute for Computational science(Proposal number hp140231).The co-author of this paper, Chit Hong Yam, is supported by the JSPS international postdoctoral research fellowship.

REFERENCES

[1] H. Aguirre, Y. Yuki, A. Oyama, T. Kiyoshi: Extending AeSeH from Many-objective to Multi-objective Optimization. *Lect. Notes Comput. Sc.*Vol 8886, pp 239-250, 2014.

[2] H. Aguirre, A. Liefooghe, S. Verel, K. Tanaka: An Analysis on Selection for High-Resolution Approximations in Many-Objective Optimization. *Lect. Notes Comput. Sc*. Vol 8672, pp 487-497, 2014.

[3] J.A. Duro, D.K. Saxena, K. Deb, Q. Zhang: Machine learning based decision support for many-objective optimization problems. *Neurocomputing*Vol 146, pp. 30-47, 2014.

[4] C. Luecken, B. Baran, C. Brizuela: A survey on multi-objective evolutionary algorithms for many-objective problems. *Comput. Optim. Appl.* Vol. 58(3), pp. 707-756, 2014.

[5] Y. Yuan, H. Xu, B. Wang: An improved NSGA-III procedure for evolutionary many-objective optimization. *Proc. 2014 Conf. Gen. Evo. Comput. (GECCO*), pp. 661-668, 2014.

[6] M. Schlueter, C.H. Yam, T. Watanabe, A. Oyama:Many-Objective Optimization of Interplanetary Space Mission Trajectories. *Proceedings of the IEEE-CEC2015 conference*, Sendai, Japan (accepted, to be published).

[7] M. Schlueter, J.A. Egea, J.R. Banga: Extended Ant Colony Optimization for non-convex Mixed Integer Nonlinear Programming. *Comput. Oper. Res*., 36(7), pp. 2217-2229, 2009.

[8] M. Schlueter:MIDACO Software Performance on Interplanetary Trajectory Benchmarks. *Advances in Space Research* (Elsevier), Vol 54, Issue 4, Pages 744 - 754 (2014).

[9] J. A. Sims, P. A. Finlayson, E. A. Rinderle, M. A. Vavrina, T. D. Kowalkowski: Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design, *AIAA/AAS Astrodynamics Specialist Conference*, Aug. 2006, AIAA 2006-6746.

[10] M. Vasile, F. Zuiani: Multi-agent collaborative search: an agent-based memetic multi-objective optimization algorithm applied to space trajectory design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*Vol 225, No. 11, pp. 1211-1227, 2011.

[11] F. Zuiani, M. Vasile: Multi agent collaborative search based on Tchebycheff decomposition. *Computational Optimization and Applications*Vol 56, No. 1, pp. 189-208, 2013.

[12] European Space Agency (ESA) and Advanced Concepts Team (ACT). Gtop database - global optimisationtra-jectory problems and solutions, Software available at http://www.esa.int/gsp/ACT/inf/projects/gtop/gtop.html, 2015.