Information Sciences 384 (2017) 34–85

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Swarm Intelligence and Evolutionary Algorithms: Performance versus speed

Adam P. Piotrowski[a,*], Maciej J. Napiorkowski[b], Jaroslaw J. Napiorkowski[a], Pawel M. Rowinski[a]

[a] Institute of Geophysics, Polish Academy of Sciences, Ks. Janusza 64, Warsaw 01-452, Poland
[b] Faculty of Building Services, Hydro and Environmental Engineering, Warsaw University of Technology, Nowowiejska 20, Warsaw 00-653, Poland

ARTICLE INFO

*Article history:*
Received 22 April 2016
Revised 14 December 2016
Accepted 17 December 2016
Available online 18 December 2016

*Keywords:*
Convergence speed
Genetic Algorithm
Differential Evolution
Particle Swarm Optimization
Biogeography-based optimization
Direct Search method

ABSTRACT

The popularity of metaheuristics, especially Swarm Intelligence and Evolutionary Algorithms, has increased rapidly over the last two decades. Numerous algorithms are proposed each year, and progressively more novel applications are being found. However, different metaheuristics are often compared by their performance on problems with an arbitrarily fixed number of allowed function calls. There are surprisingly few papers that explore the relationship between the relative performance of numerous metaheuristics on versatile numerical real-world problems and the number of allowed function calls.

In this study the performance of 33 various metaheuristics proposed between 1960 and 2016 have been tested on 22 numerical real-world problems from different fields of science, with the maximum number of function calls varying between 5000 and 500,000. It is confirmed that the algorithms that succeed in comparisons when the computational budget is low are among the poorest performers when the computational budget is high, and vice versa. Among the tested variants, Particle Swarm Optimization algorithms and some new types of metaheuristics perform relatively better when the number of allowed function calls is low, whereas Differential Evolution and Genetic Algorithms perform better relative to other algorithms when the computational budget is large. It is difficult to find any metaheuristic that would perform adequately over all of the numbers of function calls tested. It was also found that some algorithms may become completely unreliable on specific real-world problems, even though they perform reasonably on others.



## 1. Introduction

Whether one likes it or not [52], large number of metaheuristics are proposed every year [1] to solve numerical optimization problems, with Evolutionary and Swarm Intelligence algorithms receiving perhaps the most attention. As a result, a number of widely respected scientific journals have emerged, such as *Evolutionary Computation* (MIT), *IEEE Transactions on Evolutionary Computation* (IEEE), *Swarm and Evolutionary Computation* (Elsevier) and *Swarm Intelligence* (Springer). The authors of this study generally agree with the authors of [52] and do not consider such an uncontrolled increase in the number of supposedly new algorithms as a positive research trend, especially if the justification for their existence is based on em-

* Corresponding author.
*E-mail addresses:* adampp@igf.edu.pl (A.P. Piotrowski), maciejnap@gmail.com (M.J. Napiorkowski), jnn@igf.edu.pl (J.J. Napiorkowski).

http://dx.doi.org/10.1016/j.ins.2016.12.028
0020-0255/

pirical tests that are not always fair [10,37]. However, we also believe that true improvement in the field of metaheuristics is possible and would be of utmost importance for practitioners from various fields of science.

It is widely accepted that novel metaheuristics should either be developed for a specific practical problem, or comparisons between various approaches must be done on a broad range of benchmarks. However, in the first case their wide applicability, and hence the sense of using the prefix "meta" [21], may be questioned, and in the second anyone may remind that according to No Free Lunch theorems, when averaged over all possible problems the expected performance of each heuristic optimizer would be equal [63]. Nonetheless, alongside novel optimizers new sets of test problems, or new real-world applications frequently arise.

In the vast majority of papers where metaheuristics are applied to numerical optimization problems, their speed is rightly measured by the number of function calls used, not the actual consumed runtime. Runtime depends on too many factors related to specific code, programming language, software or hardware used, not to mention individual programming skills, to be considered a fair measure if one wishes to compare optimization algorithms. It is also important to discriminate between the number of function calls and a term "generation" (or similar ones) that has recently been widely used for population-based metaheuristics. In fact, within a single generation some algorithms may call objective function more frequently than others, and in the case of many modern metaheuristics the number of function calls within a generation may even vary during one run. Hence, comparing algorithms on number of generations used, instead of function calls, may lead to very unfair results [10,37] and should be avoided.

One may find, however, that in the overwhelming majority of cases metaheuristics are tested upon numerical problems with a single and fixed number of allowed function calls (MNFC) that is assumed to be representative or affordable. Such an approach is sometimes called a fixed-cost scenario, contrary to the fixed-target scenario [28] which aims to as many function evaluations as needed (with or without restarting an algorithm) to find the problem solution with the objective function value not worse than the pre-designed value. Metaheuristics suffer, however, from the exploitation versus exploration dilemma [9]. Although they should balance both processes, they are in fact rarely able to both converge quickly to a local optimum and at the same time perform a successful search within various parts of the search space. As a result, when the very popular fixed-cost scenario is used, the setting of an affordable number of function calls becomes pivotal. One may expect that the more exploitative algorithms could win competitions when the number of allowed function calls is low, whereas those more explorative should perform better when more function calls are available. The literature, however, is rather lacking in papers that address the problem of relating the performance of metaheuristics with the number of allowed function calls on a large number of versatile numerical single-objective problems, especially those from the real world.

Probably the most widely known examples of such papers are the technical reports from Real-Parameter Black Box Optimization Benchmarking (BBOB) competitions [27], the studies that follow a similar framework to BBOB [38,46], or papers that aim at parameter tuning of specified algorithms [2,16,56]. However, papers that focus on parameter tuning do not pay much attention to comparing performances of various algorithms with versatile computational budgets. The rare exception is [56], where a few SHADE [55] and CMA-ES [26] based algorithms have been found competitive for two different computational budgets. On the other hand, papers based on BBOB follow a fixed-target scenario, which means that for each problem the target-to-reach function value is determined and then the number of function calls each algorithm requires to reach such a target (with or without restarts) is counted. This approach, although surely of merit, has some disadvantages. Firstly, it assumes that the function value in the global optimum is known or may be estimated, and that the target-to-reach value may be reasonably set, which is frequently not the case for real-world problems. Secondly, algorithms that do not reach the subjectively-set target values are ranked equally poor, irrespective of objective function values that they find. Thirdly, the possibility of reaching the global optimum is not verified (as algorithms are stopped after reaching subjectively-set target value). Lastly, various metaheuristics (examples include [3,6,50,66]) that modify their control parameters during the search with respect to the number of still available function calls cannot be efficiently run if that number is unspecified. These may be the reasons that fixed-target scenarios are in fact rarely used in practice. Although the fixed-cost scenario also has drawbacks, it is somehow justified by practical need – when metaheuristics are to be applied to real-world problems, there are often clear time restrictions (that may be used to evaluate the maximum number of function calls), but setting minimum quality required is frequently very difficult without additional expert knowledge, or without performing a large number of pre-runs.

An interesting statistical approach has been recently proposed to evaluate the performance of algorithms based on the convergence speed rather than the final results [14]. But such an approach modifies the procedure of ranking algorithms, rather than searching for the relationship between the number of function calls and the relative performance of methods. Furthermore, this approach cannot help with answering the question of whether the same algorithms may be successfully used with various computational budgets, and which algorithms perform relatively better when a low or high number of function calls is available. One may also ask whether the quick convergence of a particular algorithm at some stage may be important, if, to perform a final comparison, the user has to wait anyway until the whole pre-set number of function calls is exploited. Very interesting results on the relation between the convergence speed and the performance of various optimization algorithms on Traveling Salesman Problems have been published recently in [62], showing that modern metaheuristics do not outperform local search methods. That study, however, only addressed combinatorial optimization tasks.

The present paper aims to find the relation between the relative performance of large number of metaheuristics and the allowed number of function calls in the case of numerical real-world problems. For this purpose, a fixed-cost scenario is used but algorithms are run on each problem with various MNFCs, ranging between 5000 and 500,000. Thirty three real-

**Table 1**
Summary of CEC 2011 real-world problems.

| Problem | Dimensionality | Topic |
|---|---|---|
| P1 | 6 | parameter estimation for frequency-modulated sound waves (music/engineering) |
| P2 | 30 | Lennard–Jones potential problem (physics) |
| P3 | 1 | bifunctional catalyst blend optimal control (chemistry) |
| P4 | 1 | optimal control of a non-linear stirred tank reactor (chemistry) |
| P5 | 30 | Tersoff potential function for silicon – variant 1 (physics) |
| P6 | 30 | Tersoff potential function for silicon – variant 2 (physics) |
| P7 | 20 | spread spectrum radar polly phase code design (communication) |
| P8 | 7 | transmission network expansion planning (power engineering) |
| P9 | 126 | large scale transmission pricing problem (economy/power engineering) |
| P10 | 12 | circular antenna array design (communication systems/physics) |
| P11.1 | 120 | dynamic economic dispatch – variant 1 (economy/power engineering) |
| P11.2 | 216 | dynamic economic dispatch – variant 2 (economy/power engineering) |
| P11.3 | 6 | static economic load dispatch – variant 1 (economy/ power engineering) |
| P11.4 | 13 | static economic load dispatch – variant 2 (economy/ power engineering) |
| P11.5 | 15 | static economic load dispatch – variant 3 (economy/ power engineering) |
| P11.6 | 40 | static economic load dispatch – variant 4 (economy/ power engineering) |
| P11.7 | 140 | static economic load dispatch – variant 5 (economy/ power engineering) |
| P11.8 | 96 | hydrothermal scheduling – variant 1 (economy/ hydraulics/engineering) |
| P11.9 | 96 | hydrothermal scheduling – variant 2 (economy/ hydraulics/engineering) |
| P11.10 | 96 | hydrothermal scheduling – variant 3 (economy/ hydraulics/engineering) |
| P12 | 26 | spacecraft trajectory optimization – Messenger (space missions/engineering) |
| P13 | 22 | spacecraft trajectory optimization – Cassini (space missions/engineering) |

coded single-objective non-dynamic numerical optimization algorithms from Swarm Intelligence, Evolutionary Computation, and other kinds of metaheuristics, as well as historical Direct Search methods proposed in the 1960s have been chosen to compete on 22 real-world problems from various fields of science (see Table 1) that have been used for the CEC2011 competition. Note that due to their specific nature, metaheuristics based on surrogate modeling [30] are not included. The main goal of the paper is to find out how rankings of metaheuristics depend on the MNFC, and to suggest approaches that are most promising for short and long-haul searches. A discussion is also presented that concerns to what extent the winners of the race of various algorithms may be chosen, and how such choices depend on the pre-set criteria. Although we do not address the issue of the proper computational runtime thoroughly, at the end of our discussion we provide a brief partial analysis as to whether, among specific codes we use (not the algorithms themselves), those that require more runtime lead to better results.

## 2. Methods

In this study 33 metaheuristics, summarized in Table 2, are tested on 22 numerical 1- to 216-dimensional real-world problems included in the CEC2011 set [12] (see Table 1), with six different MNFC values, set to 5000, 20,000, 50,000, 100,000, 150,000 (the value suggested in CEC2011) and 500,000. Since many of the real-world problems are computationally demanding, here let us mainly focus on the results achieved when computational budget is low. As we know [46] that metaheuristics cannot successfully compete with mathematical programming methods when the computational budget is lower than a few hundred times the problem dimensionality, we supposed that the lowest considered MNFC value is 5000. However, as we realize that some real-world problems do not require much computational effort, or some users may be willing to wait long for good results, tests are also performed with over three times larger MNFC than that suggested in [12] for CEC2011 problems. The choice of real-world problems from various fields of science is important for practitioners, and the results based on such problems are expected to be more reliable than those obtained on artificial benchmarks, which may easily lead to contradictory conclusions [14,44].

The present study focuses on metaheuristics, but a few historical Direct Search approaches [32] are also considered for comparisons. The list of 33 tested optimization algorithms (Table 2) includes methods proposed between 1960 and 2016. Codes that were obtained from other researchers or their web pages are marked with [*] in Table 2. Algorithms which codes were not obtained from their authors are used with reflection method as bounds handling technique, as in [44]. The competing algorithms' selection is inevitably subjective and reflects authors' preferences and interests. The metaheuristics used include the Genetic Algorithm [18] that was the winner of the CEC2011 competition [12] (original code submitted from CEC2011 web page http://www.ntu.edu.sg/home/epnsugan/ has been used in this study; see the Comments column in Table 2), two Direct Search methods – Nelder–Mead simplex [39] and Rosenbrock's algorithm [48], a number of Differential Evolution (DE) algorithms [13,54], selected Particle Swarm Optimizers (PSO) [17], a PSO-DE hybrid [19], a multi-algorithm [59] and a number of more recently proposed approaches, including variants of a Biogeography-based algorithm [51], Shuffled Complex Evolution with Principal Components Analysis–University of California at Irvine [8], Across Neighborhood Search [64] and Parallel Memetic Structures [6]. Note that although the vast majority of tested metaheuristics are population based (which reflects the current trends [1]), PMS and RA belong to the class of single solution methods. Among

**Table 2**
Algorithms tested. Abbreviations: DE – Differential Evolution; PSO – Particle Swarm Optimization; GA – Genetic Algorithm; D – problem dimensionality. [*] – marks algorithms which codes were obtained from authors of the source papers or relevant web pages (see Comments).

| Short name | Long name | Reference | Year | Population size | Comments |
|---|---|---|---|---|---|
| AdapSS-JADE | JADE with adaptive strategy selection | [22] | 2011 | 100 | Much modified variant of JADE [67] with an archive, based on novel concept of self-adaptation. The variant with normalized average reward method is used, as the best among four ones proposed in [22]. Note that the number of top p% individuals cannot be lower than 1. |
| ALC-PSO | PSO with an aging leader and challengers | [7] | 2013 | 20 | The initial particle velocities are set to 0. Velocities are restricted to be not larger than 50% of the bounds span. |
| AMALGAM [*] | A multialgorithm genetically adaptive method for single objective optimization | [59] | 2009 | variable, initialized with 10 when $D < 20$; 15 when $D < 40$; 20 when $D \geq 40$ | AMALGAM variant that merges CMA-ES, GA and PSO is used, as suggested and described in [59]. AMALGAM makes a number of sub-runs within time budget; for the conditions of commencing sub-run, see [59]. In each consecutive sub-run the population is twice larger, until it become 32 times larger than in the first sub-run. The code of AMALGAM has been obtained from prof. Vrugt, first author of [59]. |
| AM-DEGL | Adaptive memetic DEGL | [43] | 2013 | 5D within [10,500] | Adaptive Memetic DE variant, based on DEGL, SADE and NMA. Note that the number of local nearest neighbours is assumed to be not smaller than 6 in this study. |
| ANS [*] | Across Neighborhood Search | [64] | 2016 | 20 | ANS is a novel metaheuristic composed of elements that resemble those already known from PSO, GA or Rosenbrock's algorithm. All control parameters of ANS are set the same as in [64] apart from the one called "across search degree" that was tuned to the problem in [64] and varied between 1 and 40. In this paper "across search degree" is set to 10 for each problem. The code of ANS has been obtained from dr Wu's web page (http://guohuawunudt.gotoip2.com/publications.html). |
| ATPS-DE | JADE with adaptive population tuning scheme | [68] | 2013 | variable, initialized with 50 | Variant of JADE [67] with variable population size. In this study population size may adaptively vary within [10,500], and is initialized with 50. Note that the number of top p% individuals cannot be lower than 1. |
| CDE | Clustering-based DE | [4] | 2011 | 100 | Probably the first DE approach based on the concept of clustering. |
| CLPSO [*] | Comprehensive learning PSO | [34] | 2006 | 1D within [10,50] | State-of-the-art PSO variant. The bounds handling technique applied in the original CLPSO [34] is retained. Maximum particle's velocity has been limited to 20% of the bounds span for each coordinate; initial velocities are generated randomly within that range. The code of CLPSO has been obtained from prof. Suganthan, co-author of [34]. |
| CLPSO-DEGL | Hybrid CLPSO and DEGL approach | [19] | 2012 | 1D within [10,50] | One of hybrid DE-PSO algorithms, in which DEGL is used in each generation to update the local best positions of the particles in the swarm. Maximum particle's velocity has been limited to 20% of the bounds span for each coordinate. For CLPSO its original bounds handling technique is retained [34], for DEGL - rebounding method is used, as in [44]. |
| CoBiDE | Differential Evolution based on covariance matrix learning and bimodal distribution parameter setting | [61] | 2014 | 60 | The first out of a few DE variants proposed so far that put attention to the problem of coordinate system rotation. |
| DE | Differential Evolution | [54] | 1997 | 5D (within [10,500]) | First version of DE. The classical rand/1/bin mutation has been used; $F = 0.8$, $CR = 0.5$. |
| DEGL | DE with Global and Local neighborhood mutation operators | [11] | 2009 | 10D | DEGL variant with self-adaptive weight values is used, as suggested in [11]. Due to the specific DEGL requirements of finding large enough number of local neighbours, no limit on the maximum population size was set for this algorithm. |
| DE-SG | DE with separated groups | [42] | 2012 | value closest to 5D, but divisible by 10 and within [20,500]) | Distributed DE variant. Most control parameters are set to values specified in [42], but $PNI = 10$ and $MigProb = 1/PNI$. For this algorithm population size should be divisible by 10 and not smaller than 20. |
| EPSDE | DE with ensemble of mutation strategies and control parameters | [36] | 2011 | 50 | DE variant based on a specific concept of self-adaptation. |
| FDSADE | Differential Evolution with fitness diversity self-adaptation | [57] | 2009 | Variable, initialized with 5D (within [10,50]) | Self-adaptive DE variant. Its population size is self-adapted during run within [10,50] interval. |
| GA-MPC [*] | Genetic Algorithm with multi-parent crossover | [18] | 2011 | 90 | GA that was the winner of CEC2011 competition. The code of GA-MPC has been obtained from CEC2011 web page (www3.ntu.edu.sg/home/epnsugan/index_files/CEC11-RWP/CEC11-RWP.htm). One change to the original code has been made – if objective function is "not a number" (what may happen in a few CEC2011 problems even if solutions within a bounds are sampled) in the original code 0 had been set; in the modified code a very large number ($10^{100}$) is set in such a case. |
| GLPSO | Genetic Learning PSO | [24] | 2016 | 50 | A variant that hybridizes PSO and Genetic Algorithms-based search ideas. Velocities of particles are initialized within 20% of bounds span, and restricted within this range during run. |

(*continued on next page*)

**Table 2** (*continued*)

| Short name | Long name | Reference | Year | Population size | Comments |
|---|---|---|---|---|---|
| HCLPSO [*] | Heterogeneous Comprehensive Learning PSO | [35] | 2015 | 40 | A distributed version of CLPSO, with two sub-swarms, that aims at preserving diversity of particles within the swarm. In particular generation each sub-swarm concentrate solely either on exploration or exploitation. Velocities of particles are initialized within 20% of bounds span, and restricted within this range during run. The code has been obtained from one of co-authors of the algorithm. |
| jDElscop | Self-adaptive differential evolution algorithm using population size reduction and three strategies | [3] | 2011 | variable, initialized with min[10D,500] | Self-adaptive DE variant with the population diminishing strategy used during the search. During run the population size of jDElscop is diminished by half three times, after 25%, 50% and 75% allowed function calls is used. However, in this study the population size is not diminished if after that it would be lower than 5. |
| LBBO [*] | Linearized Biogeography-based Optimization | [51] | 2014 | 50 | A relatively new Biogeography-based optimization variant that uses additional local search technique and is able to re-initialize during run. The code of LBBO has been submitted from Prof. Simon's web page (http://academic.csuohio.edu/simond/bbo/linearized/). A change to Systematic Search procedure has been made, to prevent it from using too many function calls during run (in the original code in this procedure it was not checked whether the number of allowed function calls has been exceeded or not). |
| MDE-pBX | Modified DE with p-best crossover | [29] | 2012 | 100 | MDE_pBX introduces another DE crossover and mutation operators. |
| MPEDE [*] | Multi-population ensemble DE | [65] | 2016 | 5D, but within [50,500] | MPEDE is a new self-adaptive DE variant that bring together the benefits from both adaptive and distributed DE proposals. In the original paper [65] population size was set fixed to 250, what is, however rather too large choice for some low-dimensional problems included in CEC2011. In [65] tests were performed on 30- and 50-dimensional problems only, hence in this study MPEDE population size is related to the problem dimensionality and set to 5D (as 5•50 = 250), but within [50,500]. The code of MPEDE has been submitted from dr Wu's web page (http://guohuawunudt.gotoip2.com/publications.html). |
| NMA | Nelder-Mead simplex | [39] | 1965 | D + 1 | The variant based on [33] is used. When solving CEC2011 problems, a re-initialization approach has been used, defined as NMA1 in [44]. |
| PMS | Parallel Memetic Structures | [6] | 2013 | 1 | PMS is non-population based heuristic, partly based on RA and some DE operators. Its construction is based on Ockham's razor principle, that is so frequently violated in modern metaheuristics. |
| PSO-iw | PSO with inertia weight | [50] | 1998 | 40 | The initial particle velocities are randomly generated within 20% of the bounds span. During run velocities are also restricted to be not higher than 20% of the bounds span. |
| RA | Rosenbrock's algorithm | [48] | 1960 | 1 | The following parameter values are used: $\alpha = 3$, $\beta = -0.5$, initial step = 0.1 (variable during search). When solving CEC2011 problems a re-initialization approach has been used, defined as RA2 in [44]. |
| Rcr-JADE | Crossover-rate repaired JADE | [23] | 2014 | 100 | The variant of very popular DE method called JADE (with archive) [67], that modifies the adaptation rule used for the Crossover parameter. The control parameter adaptation methods proposed in JADE become a standard in some more recently proposed DE variants [22,25,55]. Note that the number of top p% individuals cannot be lower than 1. |
| SADE | Self-adaptive DE | [47] | 2009 | 50 | Probably the most popular adaptive DE variant. The learning period (LP in [47] has been set to 40, as a modest choice according to parameter sensitivity presented in [47]). |
| SapsDE | Differential Evolution algorithm with self-adaptive population resizing mechanism | [60] | 2013 | variable, initialized with max[1D,50] | Self-adaptive DE variant, based on the concepts from JADE with archive. Contrary to original JADE, population size is adaptively modified during run; it cannot be lower than 50, but has no upper limit. |
| SFMDE | Super fit memetic DE | [5] | 2009 | max[100, D + 1] | Hybrid DE-PSO-NMA-RA algorithm. |
| SPS-L-SHADE-EIG [*] | Success-history-based parameter adaptation DE (SHADE) with successful-parent-selecting framework, eigenvector-based crossover and linear population size reduction | [25] | 2015 | linearly reduced during run from 19D to 4 | A recent version of DE variant called SHADE [55] with linear population size reduction, eigenvector-based crossover and successful-parent-selecting framework. The winner of one of competitions held during CEC2015 meeting. The code has been obtained from CEC2015 web page (www3.ntu.edu.sg/home/epnsugan/index_files). The parameters values defined in [25] as "default" are used. |
| SP-UCI [*] | Shuffled complex evolution with principal components analysis – University of California at Irvine | [8] | 2011 | 10(2D + 1) | The variant of SP-UCI with 10 simplexes. The MATLAB code of SP-UCI has been obtained from Dr. Chu, first author of [8]. |
| SspDE | Self-adaptive DE | [41] | 2011 | 100 | Self adaptive DE variant. |

the 33 methods that we compare some are rather historical (e.g. NMA, RA), others may be considered state-of-the-art (e.g. PSO-iw, CLPSO, SADE, AMALGAM, MDE-pBX, GA-MPC) or just emerging proposals (e.g. PMS, ANS, LBBO). Not considered in this study are metaheuristics based on surrogate modeling [30]. Such approaches may be especially helpful when MNFC is low, but are often tested on other types of problems. The performance of metaheuristics often depends on their control parameters [31] and therefore in this study the control parameter values set by the authors of a particular method are almost always used (see Table 2 for details). Studying the impact of control parameters tuning on so many different algorithms and MNFC values must be considered almost unrealistic, and probably impractical. However, many control parameters of algorithms used are adaptive or self-adaptive [1], which mitigates the problem with their choice.

Every algorithm is run 30 times on each problem for every MNFC considered (as we use the fixed-cost scenario). The best (the lowest, as all CEC2011 problems aim at minimization) function values found in particular runs are stored. All runs are independent from one another and start from a randomly initialized population (runs with larger MNFC are not the continuation of the ones with lower MNFC). This is important, as some algorithms (like jDElscop or PSO-iw) vary their control parameters during one run depending on how many function calls are still available. Consequently they would behave differently during for example the first 5000 function calls when the search may take a much longer time, than in case when only 5000 calls are available. Owing to the fact that runs with each computational budget are performed independently, it may by chance occur that better results are achieved with a lower number of allowed function calls than with a higher number; this may occur especially on multimodal problems with the largest considered MNFC values, when algorithms are endangered with either premature convergence or stagnation.

Two approaches to discuss the results are proposed in this paper. Firstly we consider rankings of algorithms averaged over 22 problems. We then shift our attention to the quantity and types of specific problems that particular algorithms succeed in, when compared to others, or when they are among the best methods in each out of six competitions.

In the first case, the general relation between the rankings of algorithms averaged over all 22 problems and the MNFCs is to be found and discussed. Averaged rankings are computed as follows. Initially all algorithms are ranked independently for each problem and computational budget. The best function values found during each run by the specific algorithm with a particular MNFC budget are averaged over 30 runs. Then algorithms are ranked from the one with the lowest 30-run averaged objective function value (which is considered the best and receives rank 1), to the one with the highest 30-run averaged objective function value (rank 33 – the worst). In [44] it was noted that using 30-run average or median does not change the ranking of algorithms significantly, therefore we use only the rankings based on the mean performance in this study. It is worth noting that in the case of some real-world problems, the results obtained by various algorithms may differ just marginally, which would have no practical meaning, and therefore in such cases it is inappropriate to assign various ranks to almost equally performing methods. Hence in this paper it is assumed that when the difference between the 30-run averaged objective function values achieved by some algorithms is smaller than a pre-defined very small value (set to $10^{-10}$ here) for a particular problem, such algorithms are considered to perform equally well and receive equal rank (for example, rank 1.5 if such a low difference is noted between only the two best ones). Finally, ranks achieved by each algorithm for a particular MNFC are averaged over 22-problems, to verify the problems-aggregated performance on a particular computational budget. Such 22-problems averaged rankings facilitate the discovery of some relations between the overall performance of algorithms and the number of allowed function calls. However, as the specific performance of algorithms on particular types of problems are thus lost, the second part of the discussion is based on the statistical significance of the differences between the results obtained by various metaheuristics, and on the graphical illustration of the results obtained by each algorithm on particular problems with each considered MNFC (see Figs. 1–22). We search for algorithms that perform much better, or much worse, on various problems than the majority of the flock. To do so, apart from visual comparison we verify for how many problems (and, if possible, for what kind of problems) the particular algorithm performs best, or is ranked one of the top three, or one of the top five best methods when a specific computational budget is set. We also count how many times the differences between that particular algorithm and other metaheuristics are statistically significant. In this way we may find some more specialized methods, as well as those that often perform similarly well (or poorly). The statistical significance of pairwise comparisons among all 33 algorithms is tested by means of the Friedman's test with post-hoc Shaffer's static procedure [49] at $\alpha = 0.05$, as suggested in [20,58] for experiments where a fair comparison of a number of already existing methods is needed. The respective codes of statistical tests were obtained from www.cmpe.boun.edu.tr/~ulas/m2test/details.php [58].

It is important to note that various metaheuristics may have different population sizes. If the population size used is very large, this may hinder the verification of the performance of such a method with the lowest computational budget, which has been pre-set to 5000 function calls. This is, for example, the case for SP-UCI with 10 complexes tested on 216-dimensional problem F11.2. It emerges that the population size of SP-UCI with 10 complexes for the 216-dimensional problem is as large as $10 \bullet (2D + 1) = 4330$, when $D$ is the problem dimensionality. Hence it requires more than 5000 function calls to initialize and perform at least one generation. It is difficult to test the algorithm if it is unable to finish at least one generation, but doing so uses over 8000 function calls, which may make the comparison unfair. However, as SP-UCI does not perform well in competition based on the smallest computational budget (as will be seen in Section 3, it is ranked the worst method when MNFC is set to 5000), we allowed it to use such extra calls. A similar problem is encountered with SPS-L-SHADE-EIG, which has initial population size set to $19 \bullet D$, with this population size gradually decreasing during the run. However, according to the code that is freely available from the CEC2015 site (www3.ntu.edu.sg/home/epnsugan/index_files), SPS-L-SHADE-EIG prematurely terminates when the maximum number of function calls is to be exceeded during the partic-

ular generation, if current population size is to be used. As this "official", widely-available code would probably be used by the majority of other users, contrary to the case with SP-UCI, we decided to not modify the code and allow the algorithm to terminate earlier on some large-scale problems when MNFC is set to 5000 (the problem does not occur when the computational budget is set to 20,000 or larger). Although this leads to poor ranking of SPS-L-SHADE-EIG on such problems, we verified that the performance of SPS-L-SHADE-EIG would improve only slightly if we allow it to perform the next generation using extra calls. The above discussion shows that the comparison of versatile algorithms, especially those that start with large populations, may be considered slightly controversial when MNFC is very low. Nonetheless, this is of limited practical importance, as all algorithms that use large population size perform poorly when the number of function calls is small. Addressing the obvious question how much their performance could be improved by limiting the initial population size is left for further research.

Finally, in sub-Section 3.3 we also provide a very brief discussion regarding the actual computational runtimes of specific codes that we use (measured in seconds) and their performances. Although we are aware that such results may be altered if one uses another machine, programming language, or even writes similar code independently, such a test may be of some interest for practitioners. They may seek suggestions as to which codes (especially those that are freely available in the web or from authors on request) require more computational time and which would lead to better results. Our computational runtime tests require that all codes are run one by one on the same machine that is not used for any other purposes (to achieve this, we used a rather old computer with Intel Core 2 Quad Q9550, 2.83 MHz, 2GB RAM, with Microsoft Windows XP 2002; codes are run in MATLAB 2012a version). Hence in this study such tests are performed only for three selected, quickly computable problems (P1, P8 and P11.6) with MNFC set to a low value of 20,000 (we do not consider MNFC equal to 5000 due to the obstacles discussed in the previous paragraph). Each code is run 30 times on each of the three problems and the mean and standard deviation of the true computational runtime in seconds is noted.

## 3. Results

The results achieved by every algorithm on each among the 22 problems are illustrated in Figs. 1–22. Each figure presents the results obtained by all 33 algorithms (arranged alphabetically from left to right) on a particular problem within six numbers of allowed function calls, from the lowest (set to 5000, top sub-figure) to the largest (set to 500,000, bottom sub-figure). Figures show, for each algorithm and computational budget, 30-run averaged objective function values (black line), the ranges of function values obtained within 30 runs with the two best and two worst runs omitted (blue shaded area), and the function values obtained in two best and two worst runs (red crosses). All detailed results are available online as Supplementary Material.

In Tables 3–8 the rankings of algorithms for each problem and computational budget are given (rankings are based on the 30-run averaged performance). Table 9 shows the ranks averaged over all 22 problems for each computational budget. For the convenience of the reader Table 10 reveals the overall placing achieved by each algorithm in every competition, which is based on averaged ranks given in Table 9 (the algorithm with the lowest averaged rank given in Table 9 receives place 1 in Table 10, the algorithm with the second lowest average rank – 2, and so on). To verify how often a particular algorithm is among the best in each competition, Table 11 provides the number of problems for which particular algorithm wins for each computational budget, the number of problems for which particular algorithm is among the best three or the best five methods, and the correlation coefficients between ranks achieved by a particular algorithm on all problems when MNFC values are set to 5000 and 150,000 (the value suggested in CEC2011 [12]), or to 150,000 and 500,000. Tables 12–17 show for each computational budget the statistical significance of the pair-wise differences between the results achieved by all algorithms. Finally, in Table 18 it is shown, for each computational budget, from how many competing algorithms the particular algorithm is significantly better than, and for how many algorithms it is significantly worse. This is rather intuitive, not formal, as statistical tests only show whether the difference between the two methods is significant at particular $\alpha$, not which method is better or worse. However, we may informally compare the mean rankings of two methods, and if the difference between the performances of both methods is statistically significant, we call the approach with the lower averaged rank significantly better, and the algorithm with the higher averaged rank significantly worse.

### 3.1. Relation between 22-problem averaged rankings and numbers of allowed function calls

As one may find from Tables 9 and 10, vastly different algorithms are ranked best when computational budget is low, than when it is high. When the MNFC is set to 5000 PSO-iw, CLPSO-DEGL, ATPS-DE, HCLPSO and ALC-PSO are the five best algorithms (their 22-problem averaged ranks vary between 7.9 for PSO-iw to 10.3 for ALC-PSO – see Table 9), but three of them are among the poorest methods when MNFC is set to 500,000: PSO-iw in this case is ranked 31st best, CLPSO-DEGL – 24th and ALC-PSO – 28th. Only ATPS-DE performs relatively well even with the largest computational budget, when it is classified as the 8th best metaheuristic, and meanwhile, the performance of HCLPSO is moderate (14th).

A similar observation is true for the opposite relationship – among the five best methods (SPS-L-SHADE-EIG, CoBiDE, AM-DEGL, MPEDE and Rcr-JADE) when MNFC is set to 500,000 (their 22-problem averaged ranks vary between 6.6 for SPS-L-SHADE-EIG and 10.4 for Rcr-JADE – see Table 9) all are ranked among the poorer half in competition with MNFC set to 5000: SPS-L-SHADE-EIG is ranked 25th, CoBiDE – 28th, AM-DEGL – 22nd, MPEDE – 21st and Rcr-JADE – 19th.

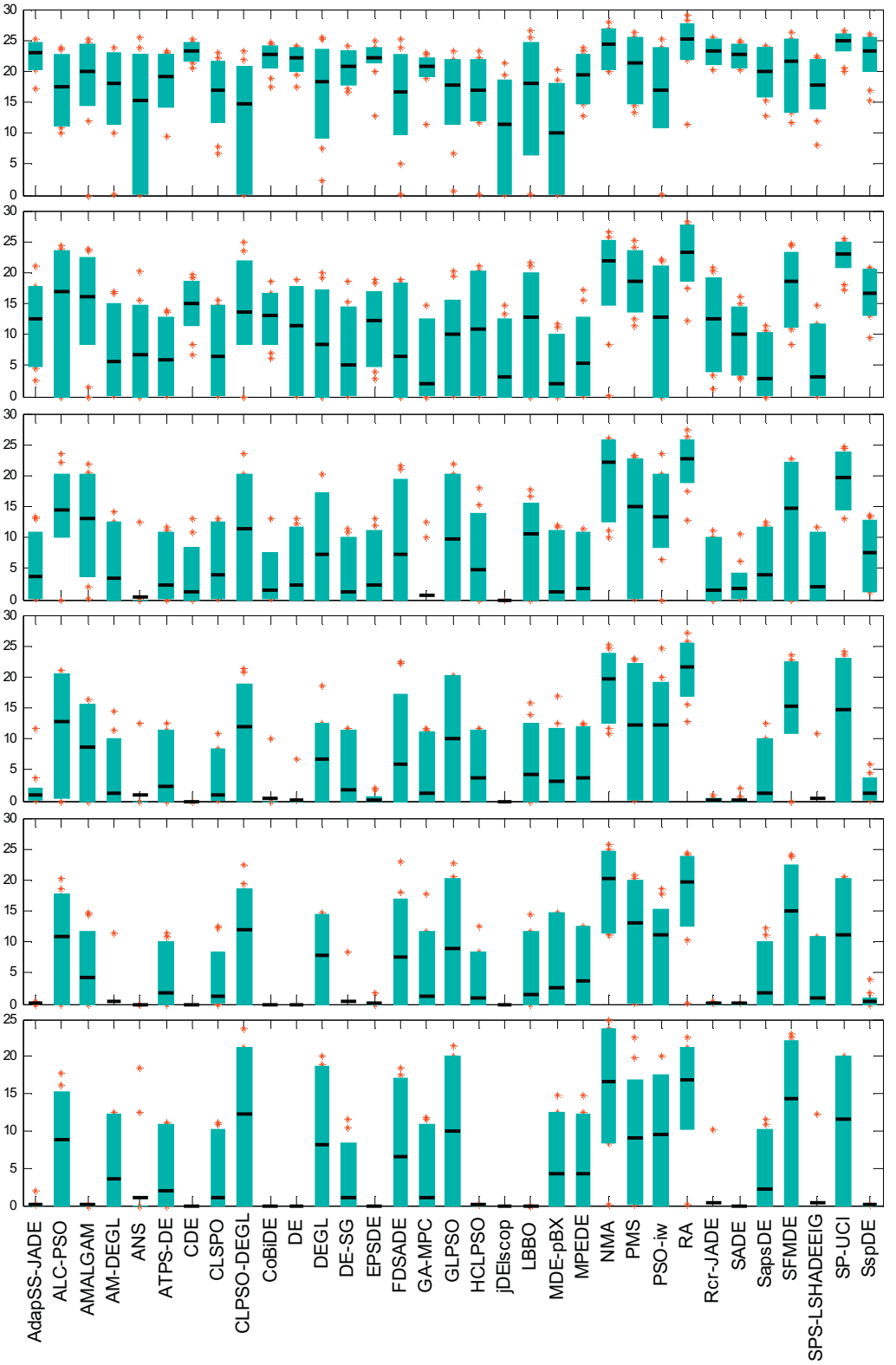


**Fig. 1.** Problem P1. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

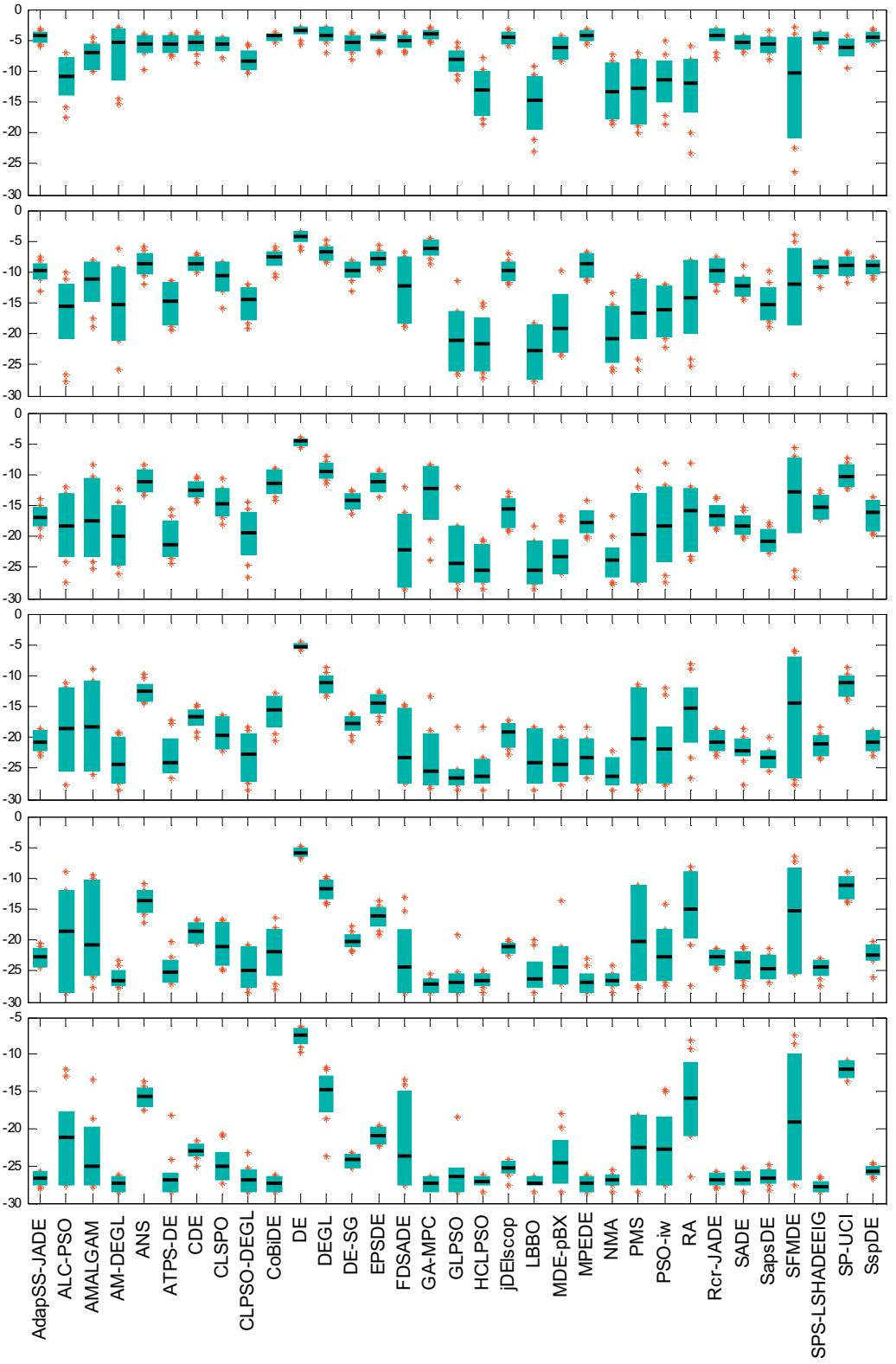


**Fig. 2.** Problem P2. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

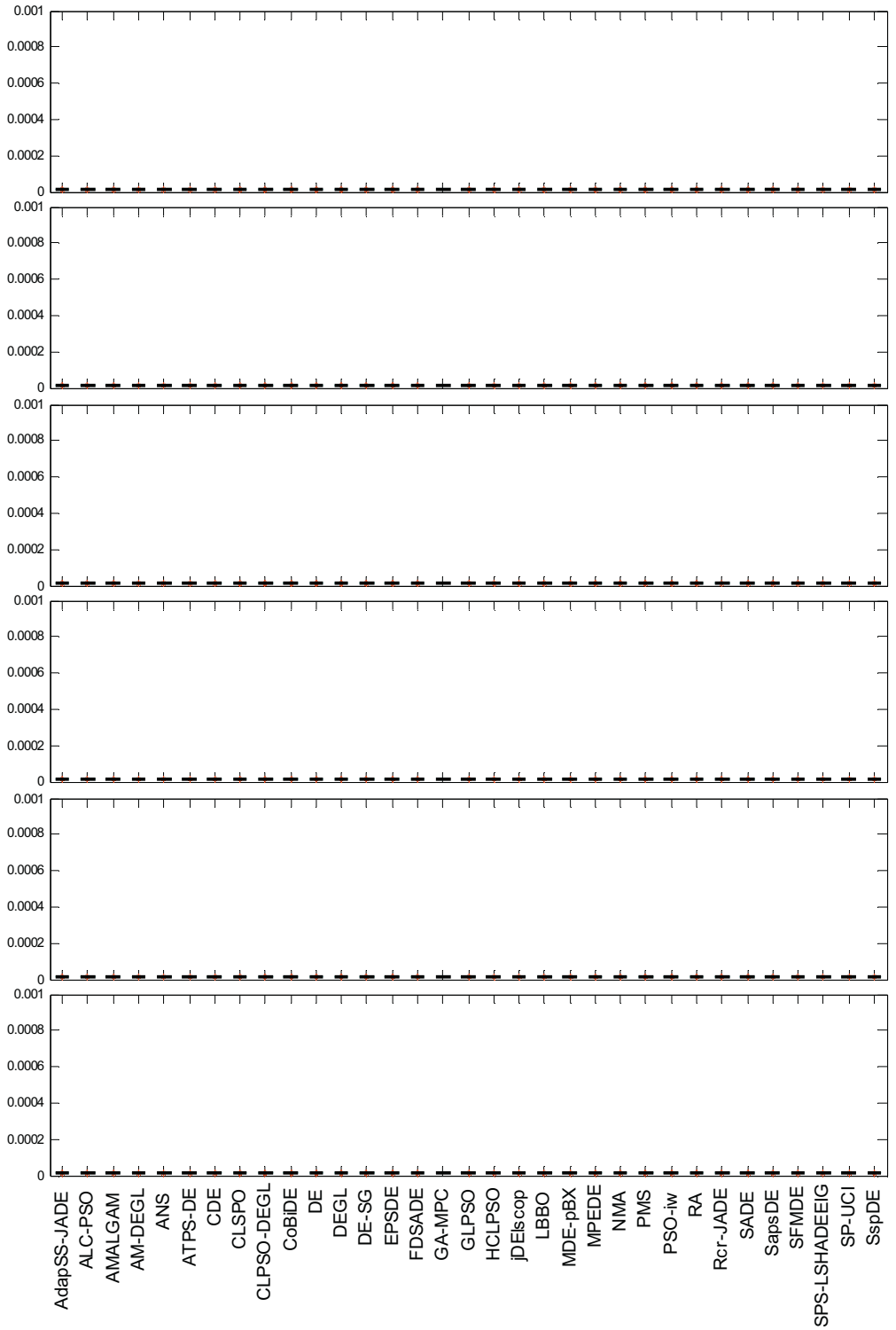


**Fig. 3.** Problem P3. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)


**Fig. 4.** Problem P4. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

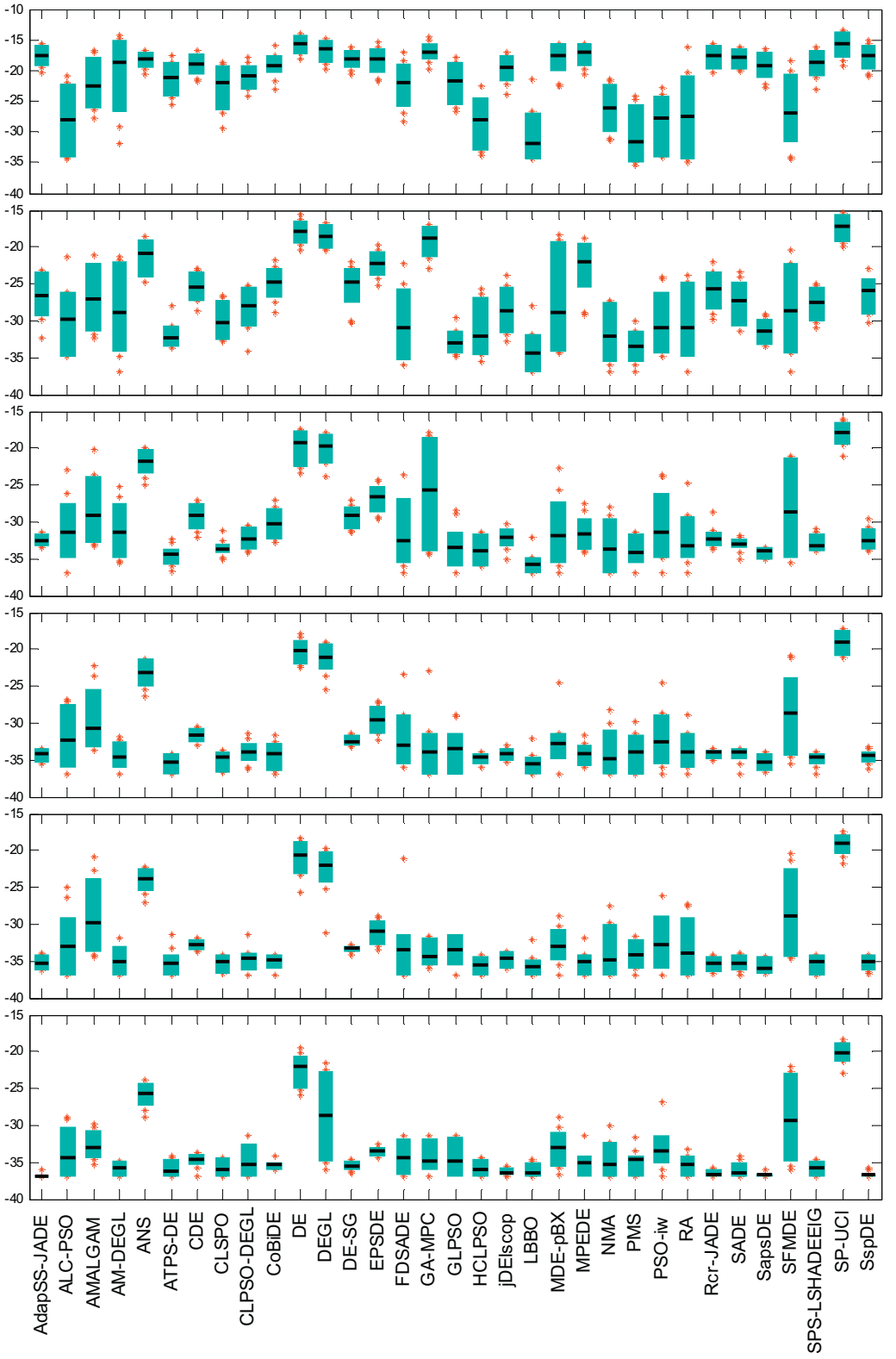


**Fig. 5.** Problem P5. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

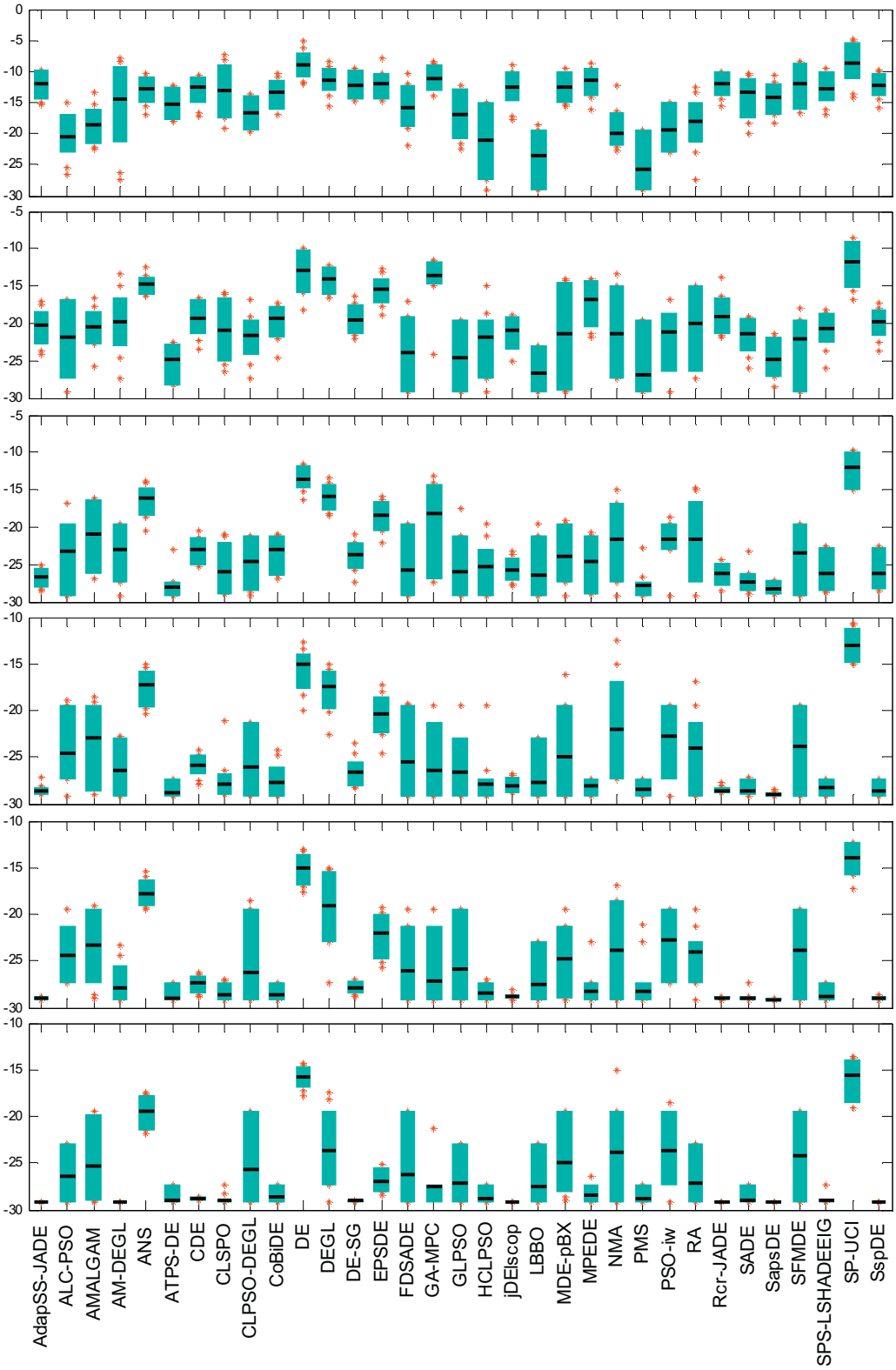


**Fig. 6.** Problem P6. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

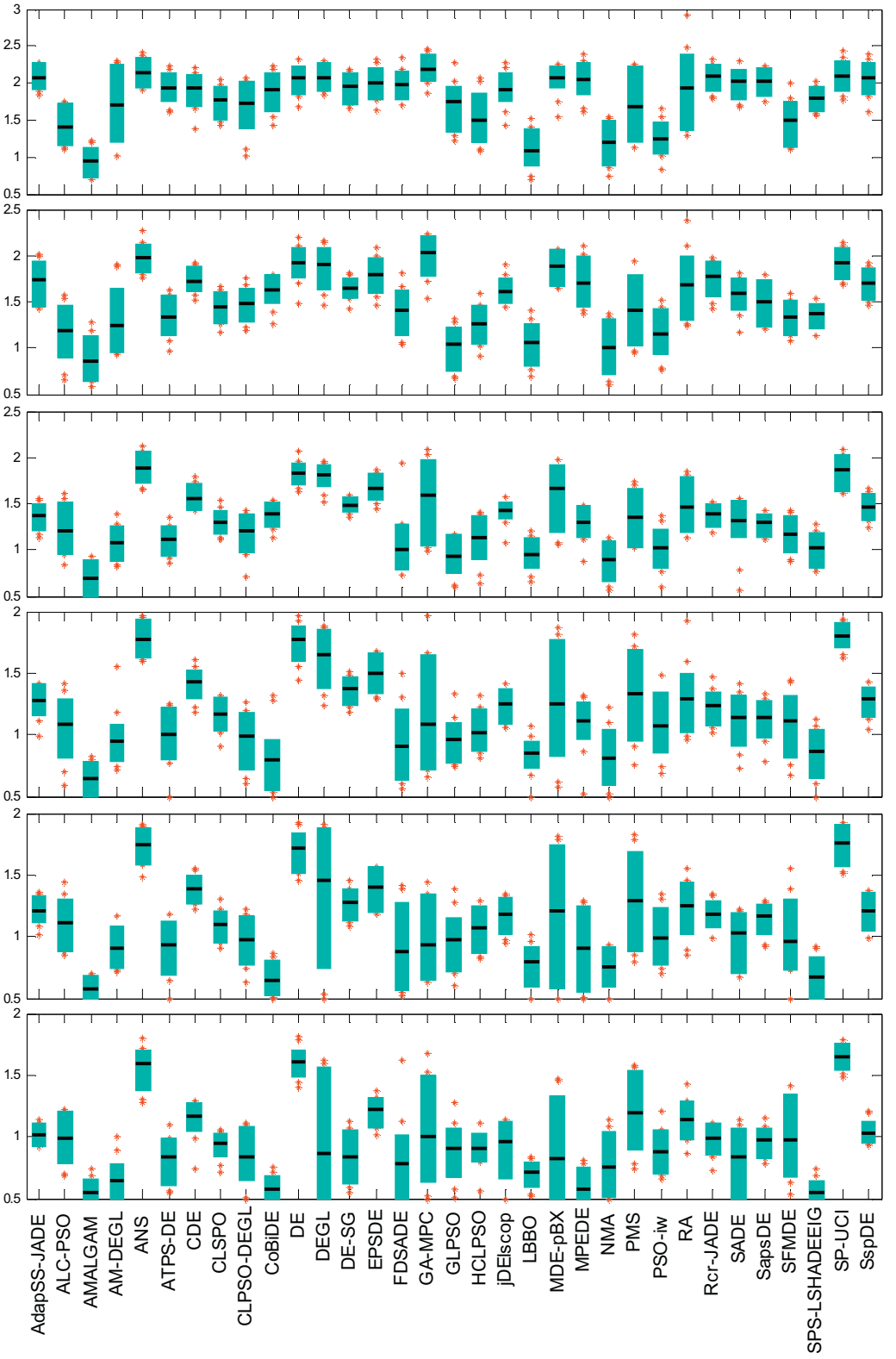


**Fig. 7.** Problem P7. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

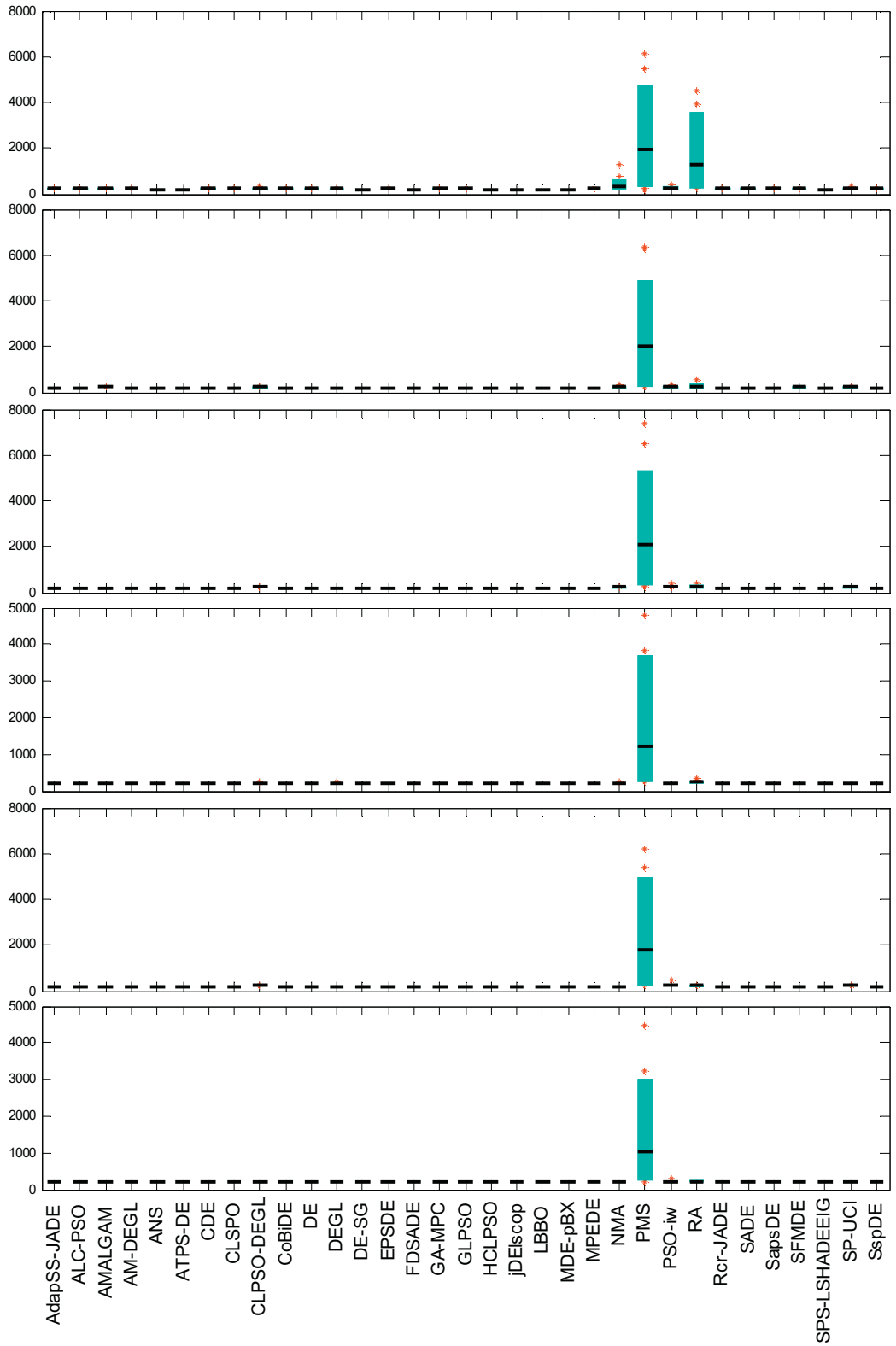


**Fig. 8.** Problem P8. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

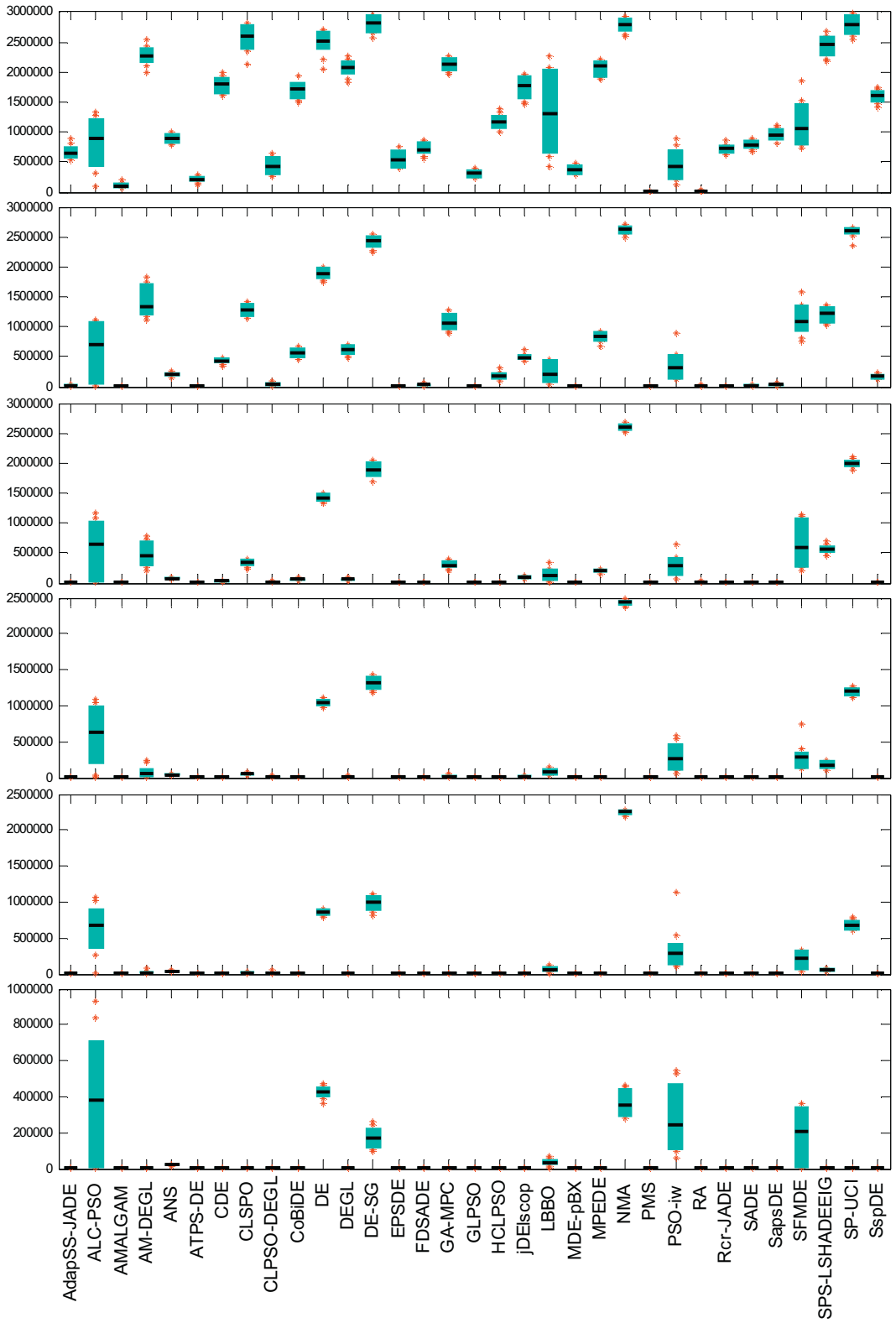


**Fig. 9.** Problem P9. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

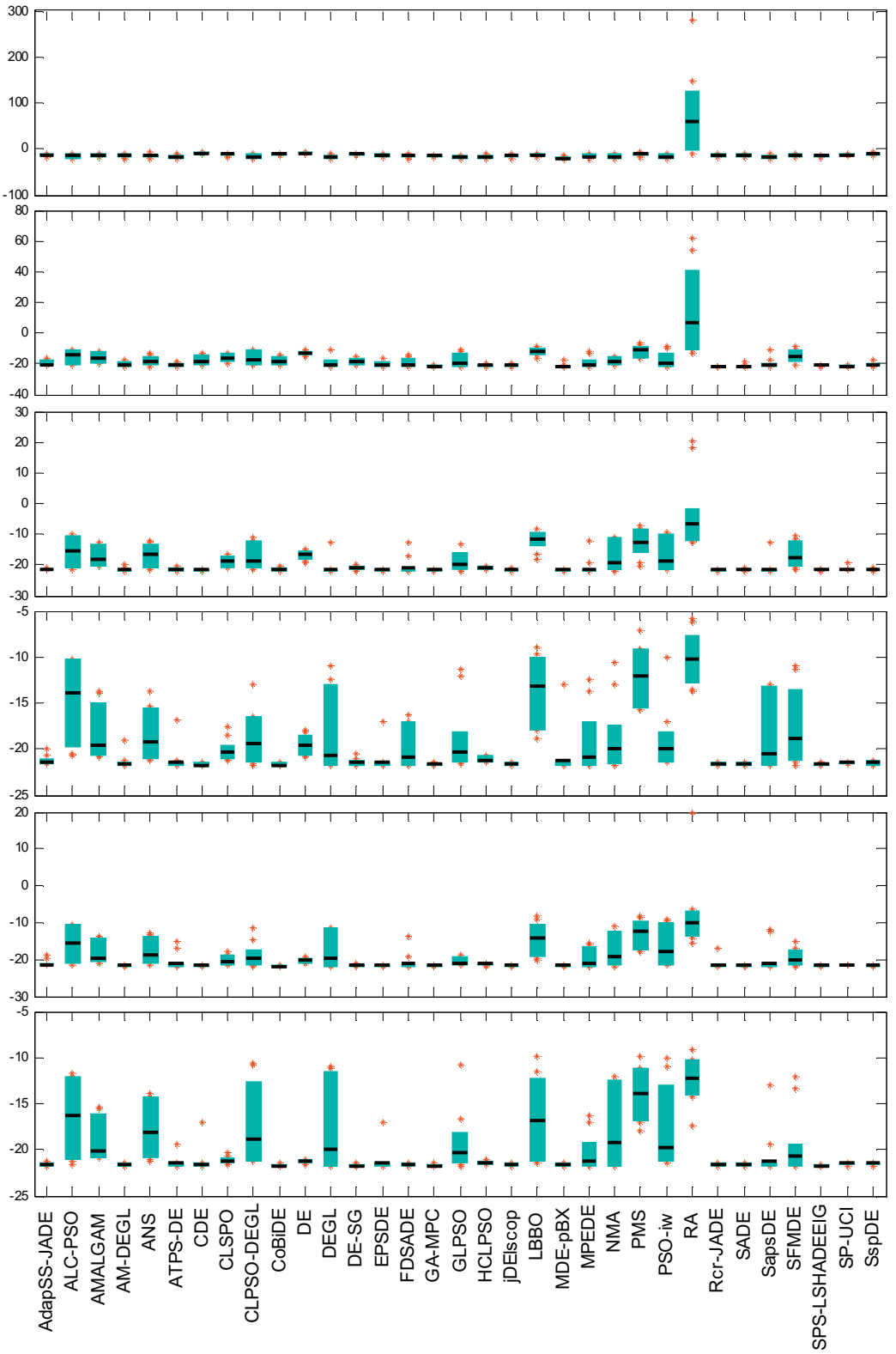


**Fig. 10.** Problem P10. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

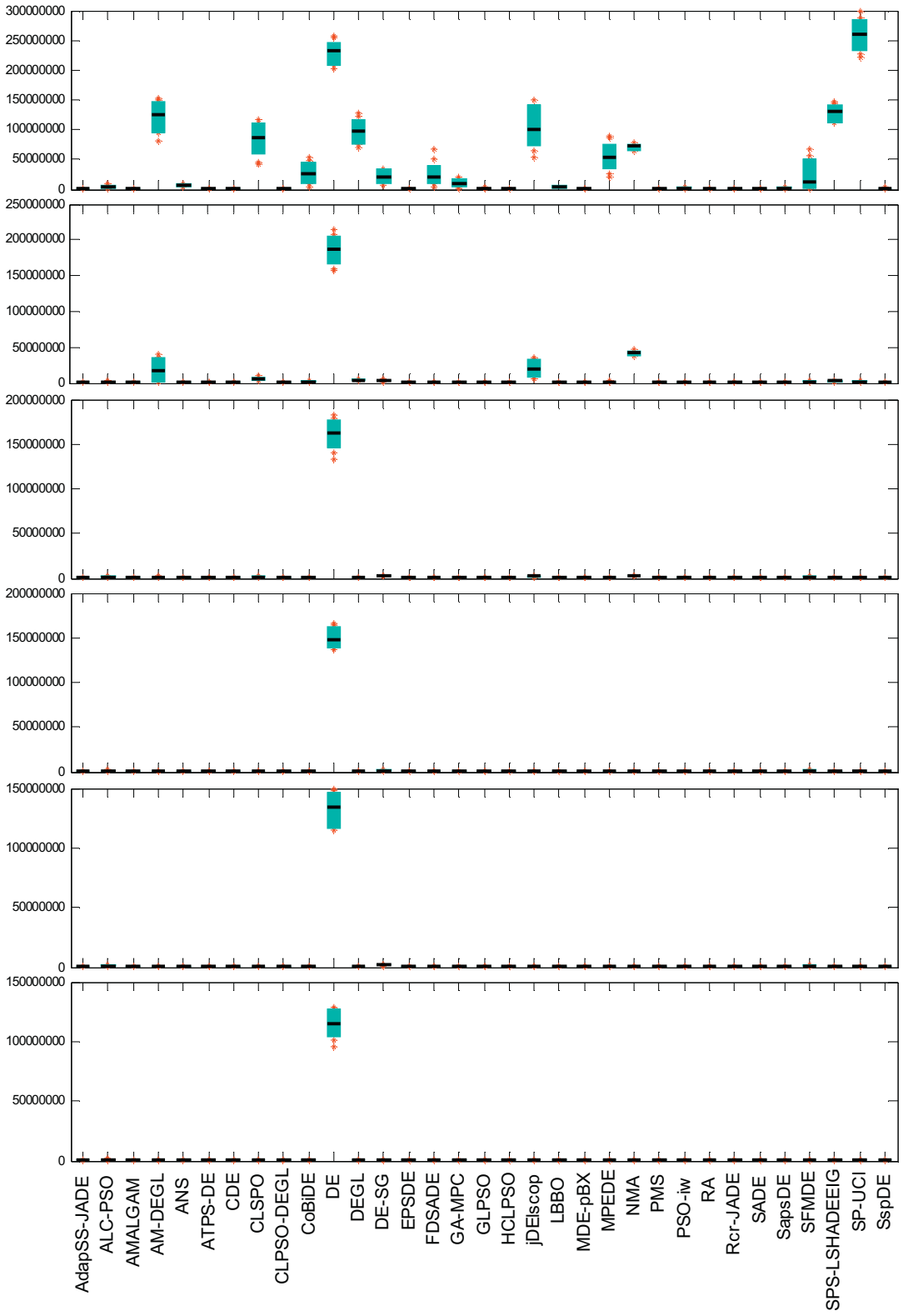


**Fig. 11.** Problem P11.1. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

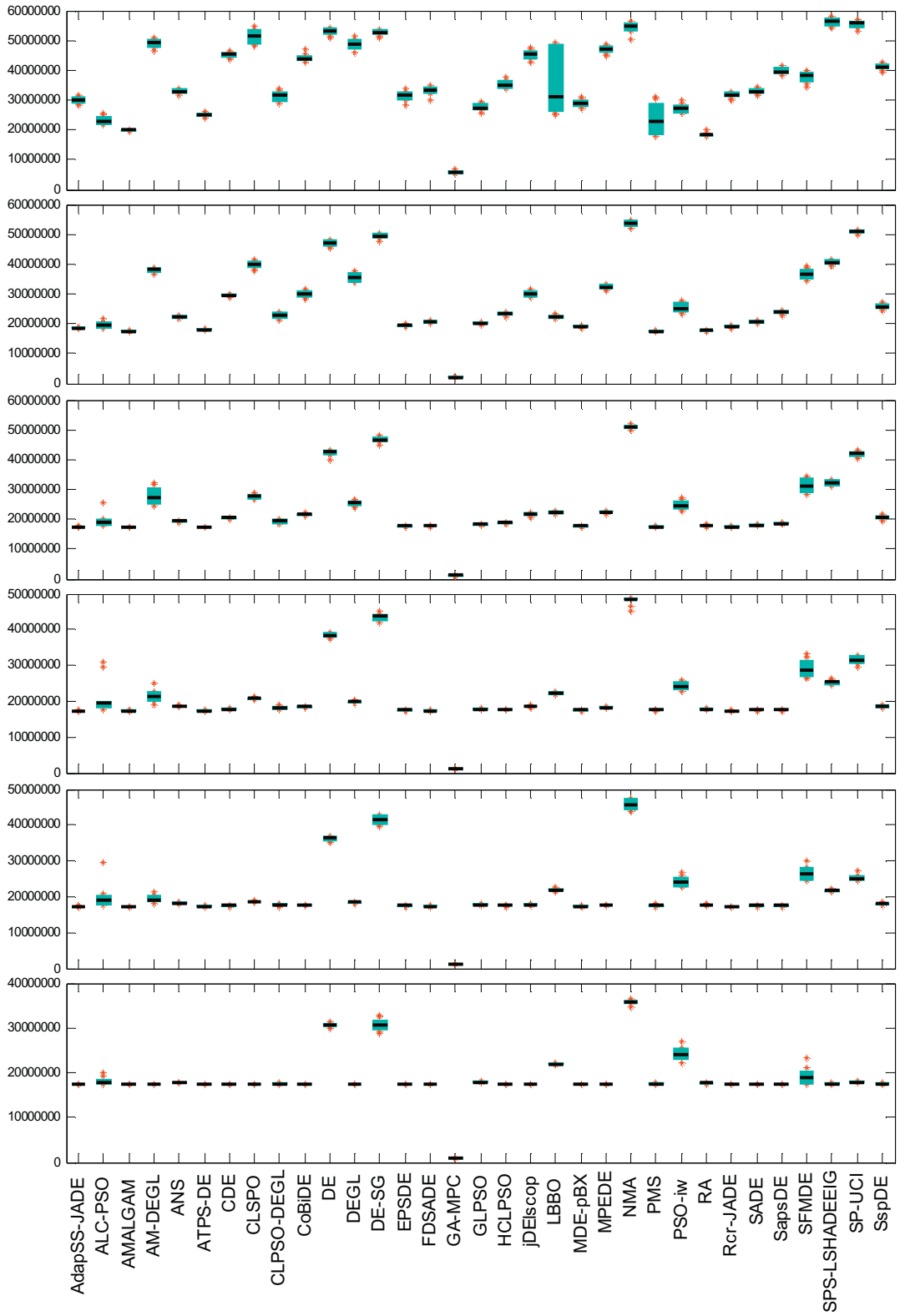


**Fig. 12.** Problem P11.2. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

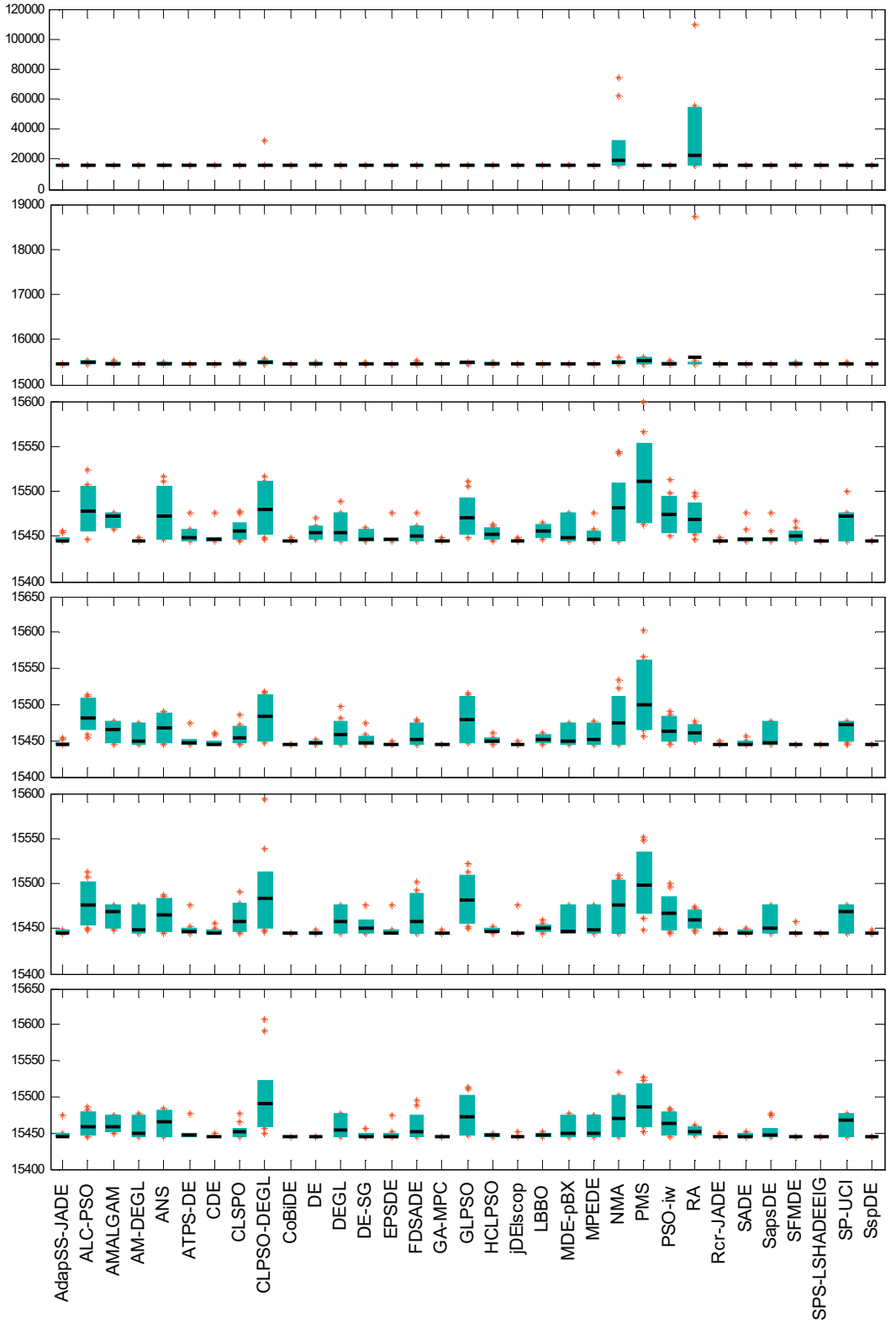


**Fig. 13.** Problem P11.3. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

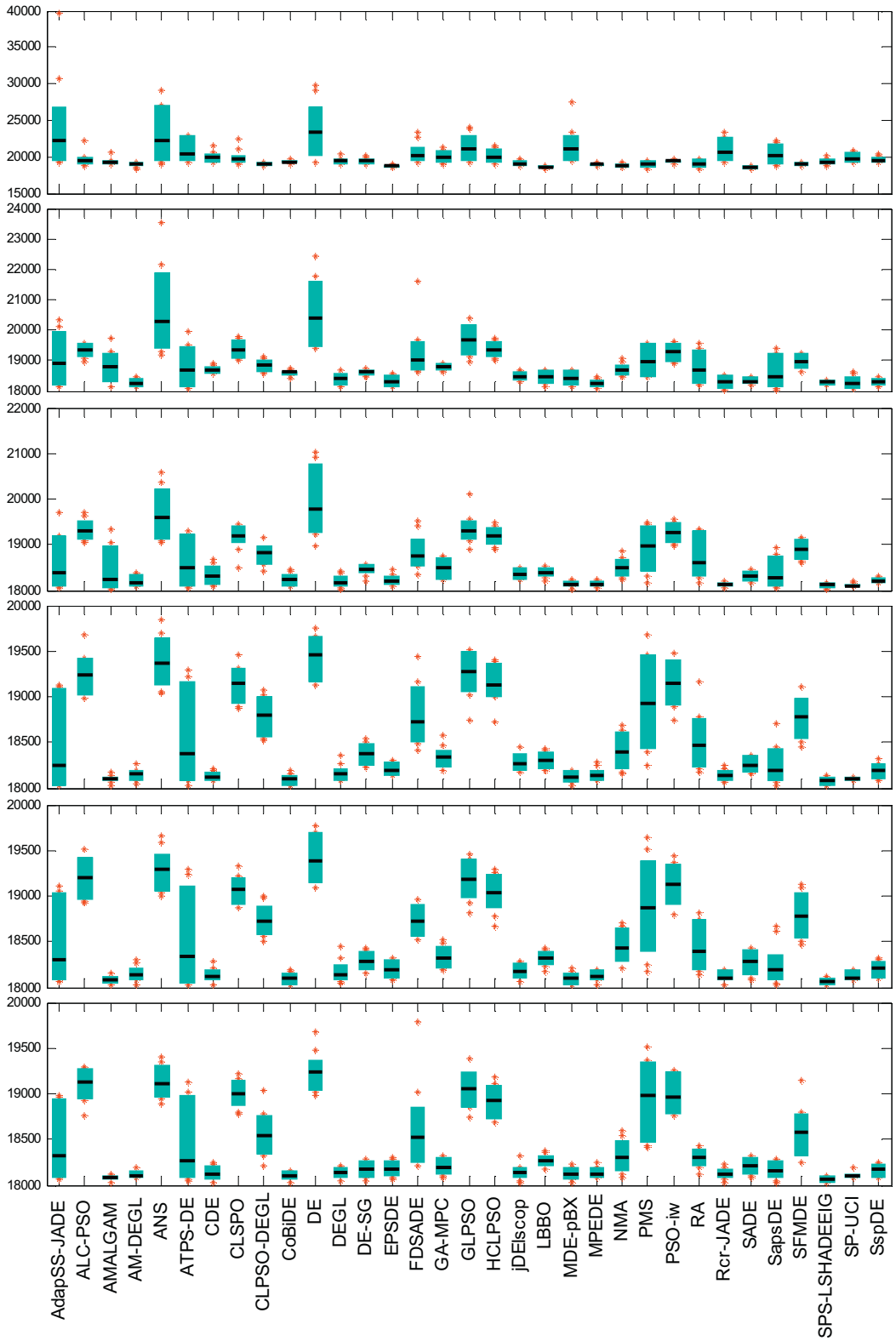


**Fig. 14.** Problem P11.4. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

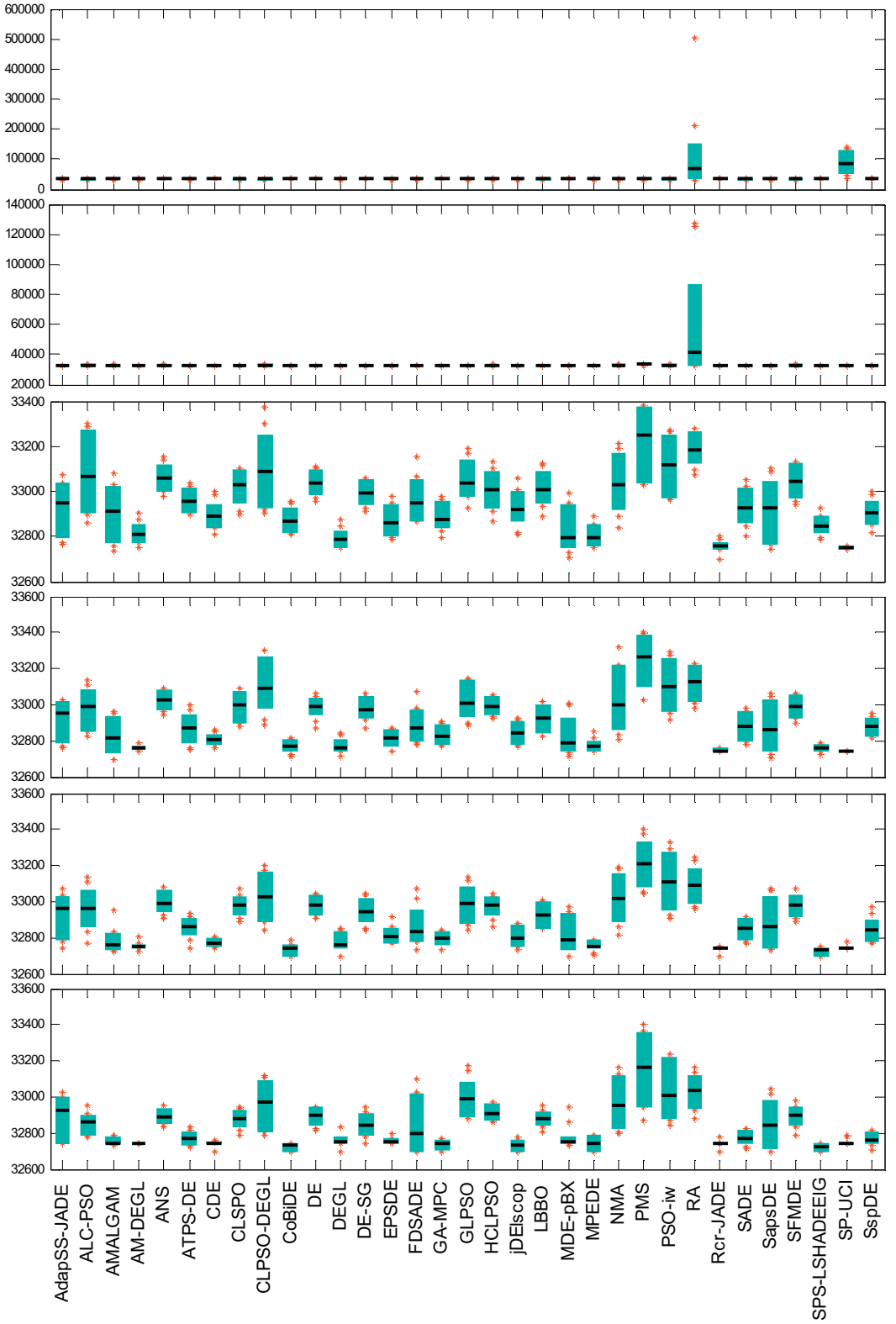


**Fig. 15.** Problem P11.5. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

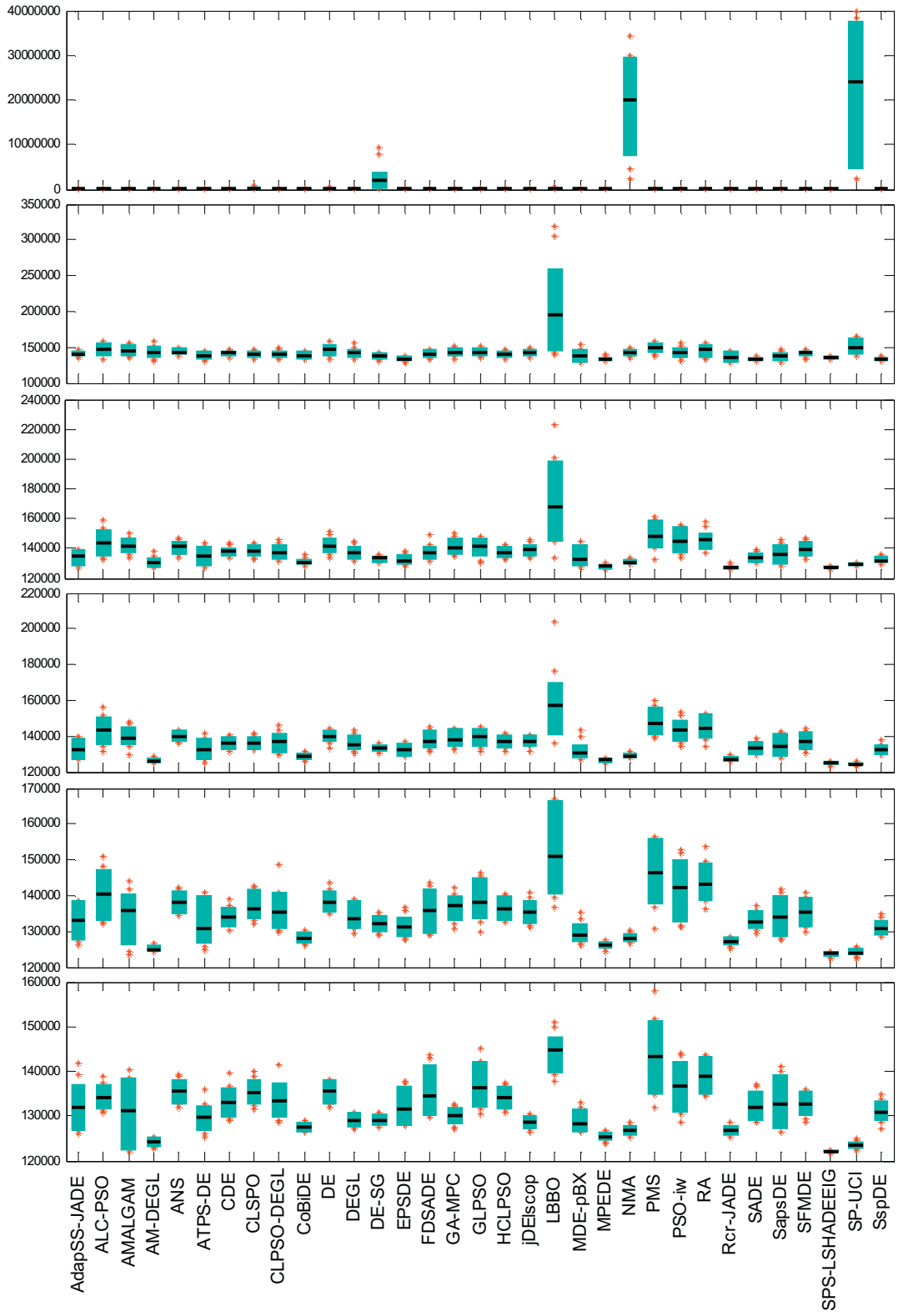


**Fig. 16.** Problem P11.6. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

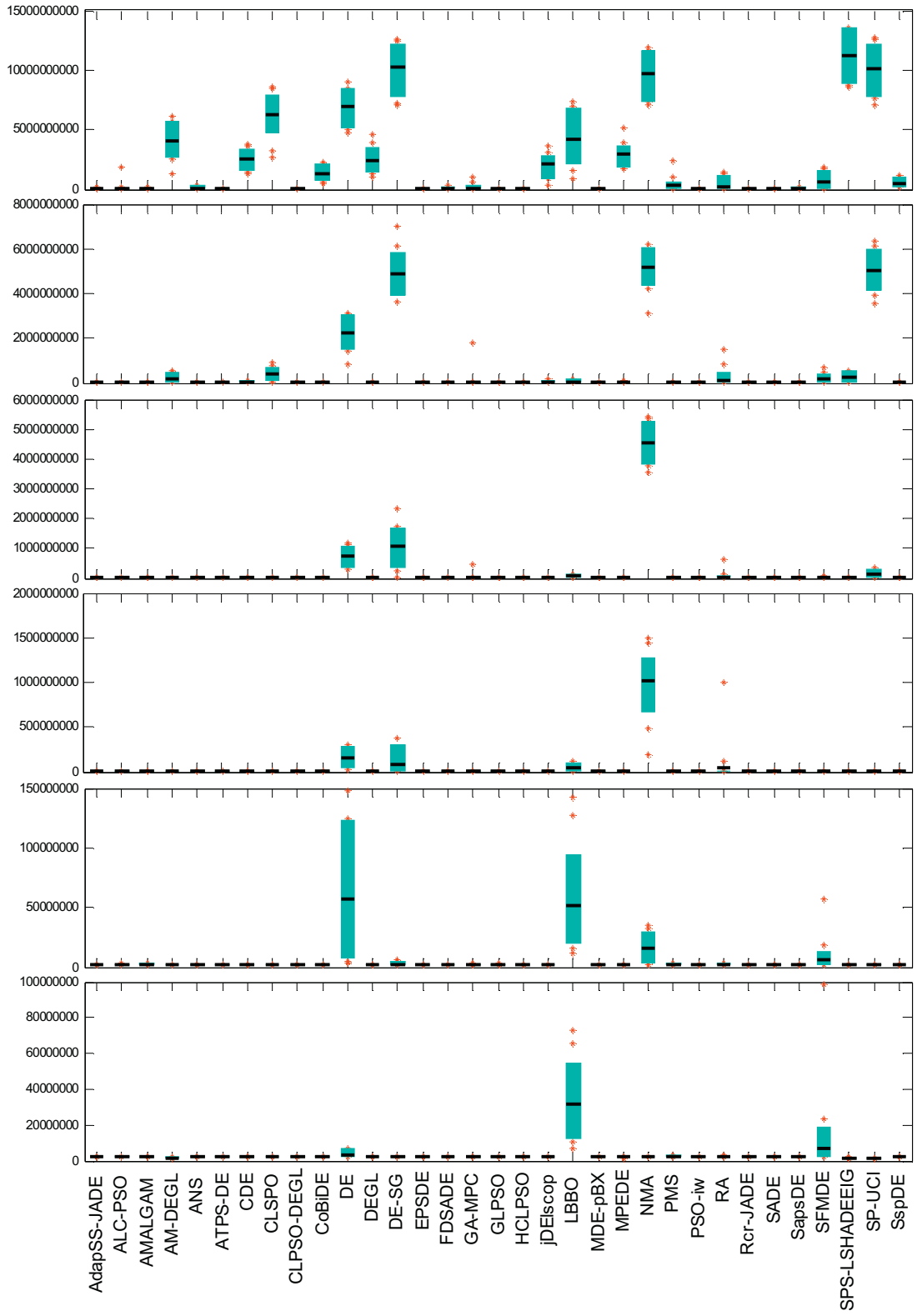


**Fig. 17.** Problem P11.7. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

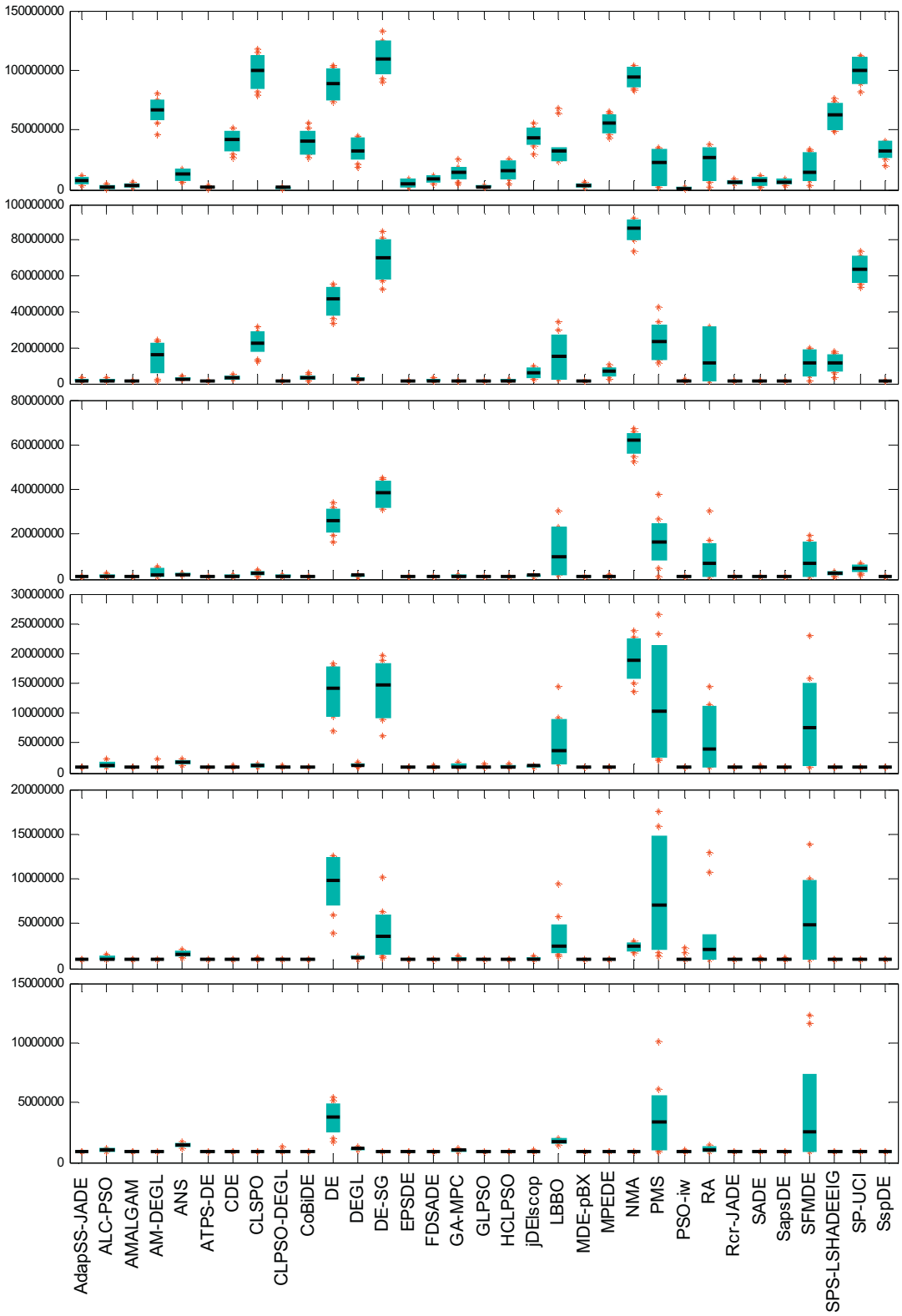


**Fig. 18.** Problem P11.8. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

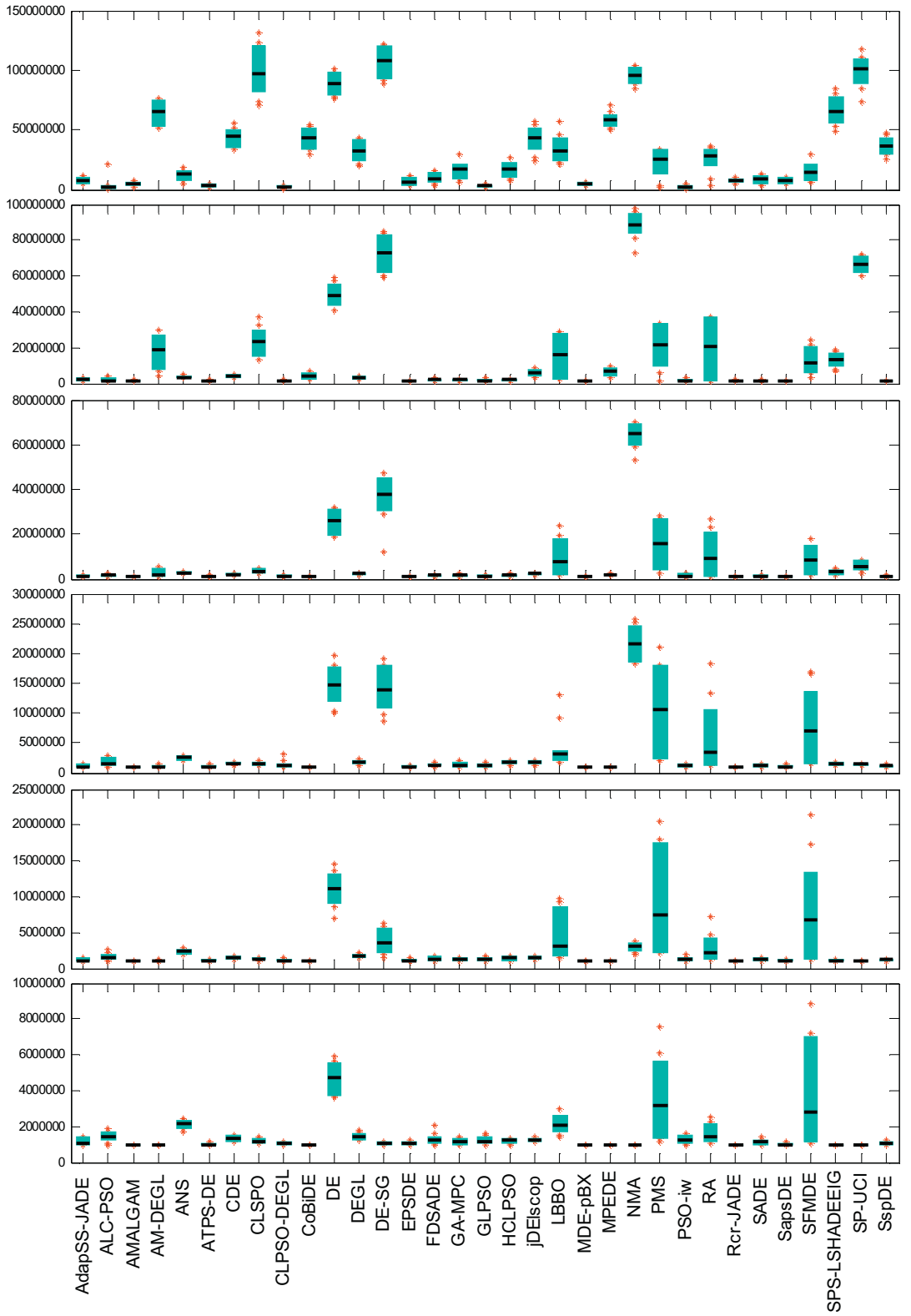


**Fig. 19.** Problem P11.9. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

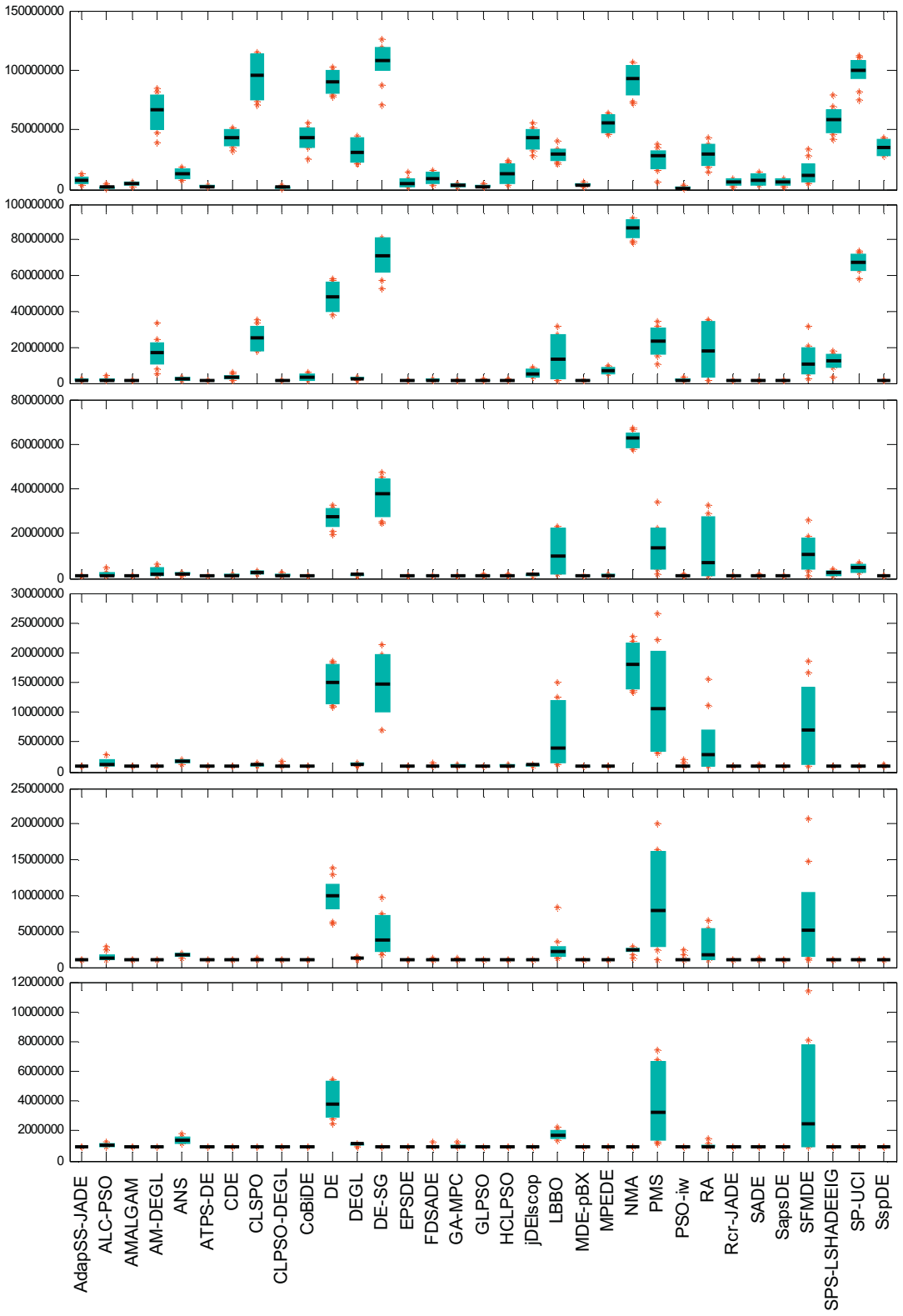


**Fig. 20.** Problem P11.10. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

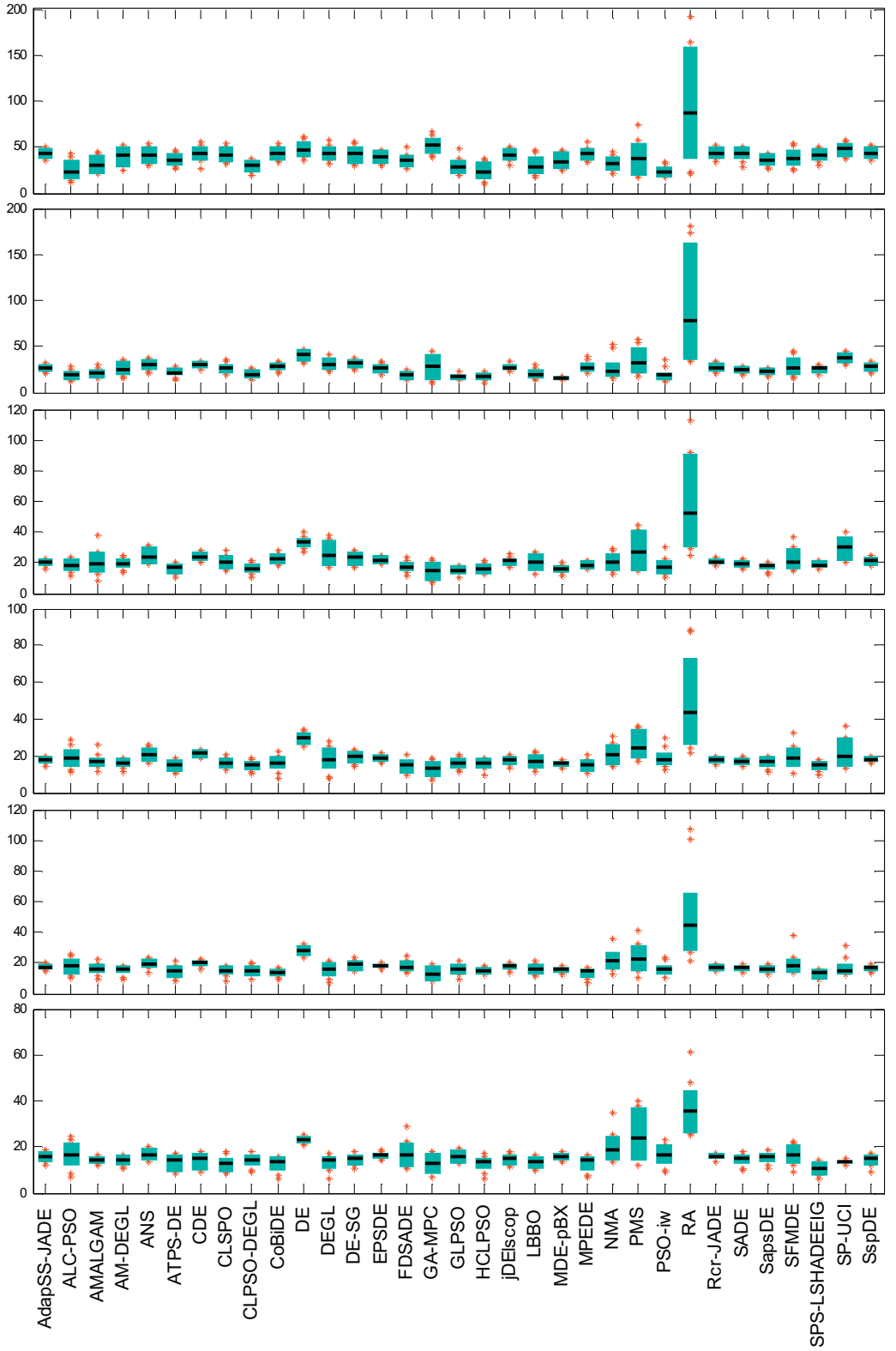


**Fig. 21.** Problem P12. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

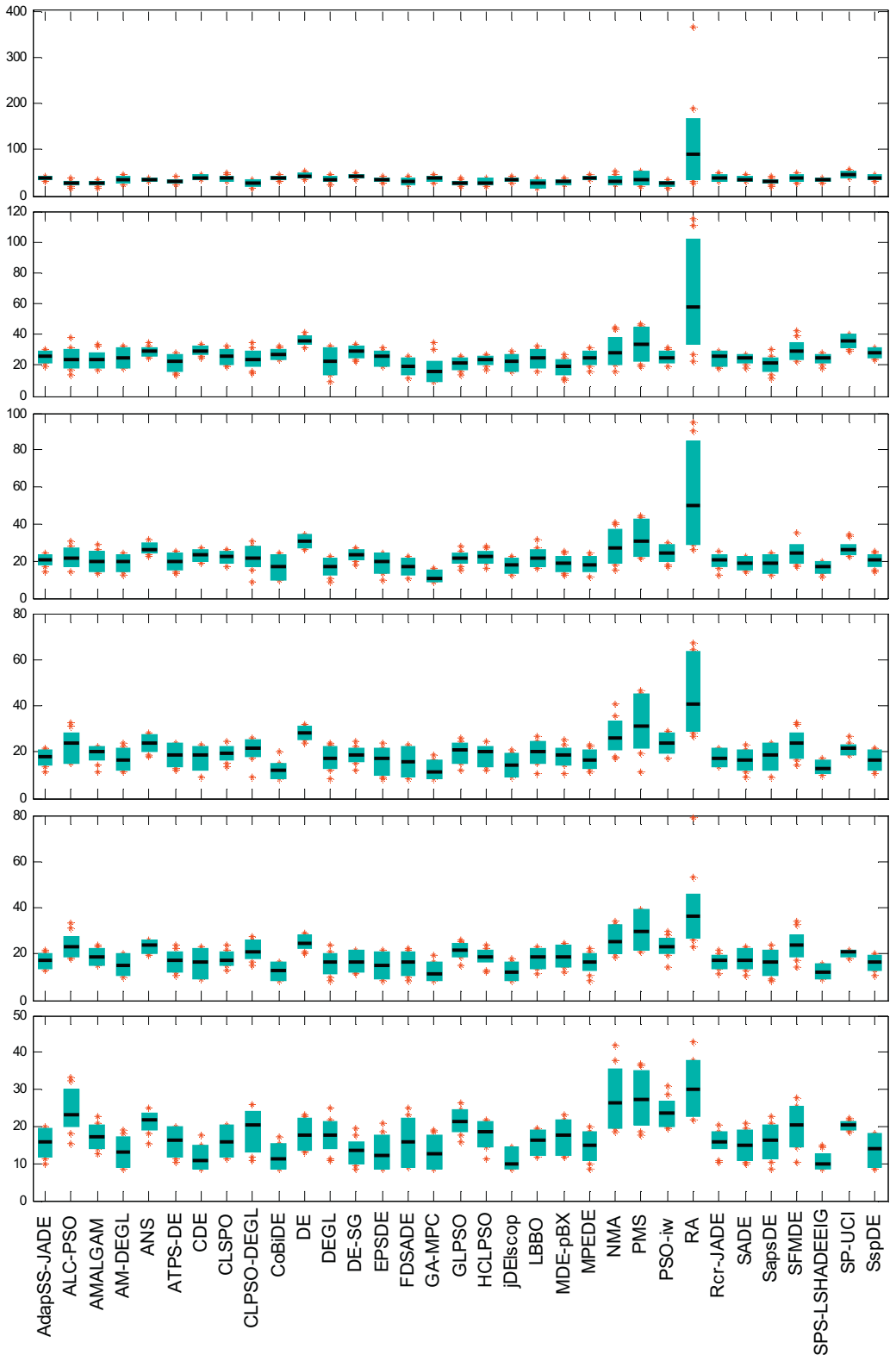


**Fig. 22.** Problem P13. Results achieved by every algorithm with MNFC set to 5000 (top subplot), 20,000, 50,000, 100,000, 150,000 and 500,000 (bottom subplot). Black lines indicate the 30-run averaged performance, the range of the results obtained from 3rd best to 28th best run is given in blue, the two best and two worst performances during 30 runs are indicated as red diamonds. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Table 3**
Ranking of algorithms on all problems with allowed number of function calls set to 5000. Algorithms with ranks 1–3 are bolded.

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11.1 | P11.2 | P11.3 | P11.4 | P11.5 | P11.6 | P11.7 | P11.8 | P11.9 | P11.10 | P12 | P13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AdapSS-JADE** | 27 | 29 | 17 | 12 | 26 | 25 | 29 | 16 | 10 | 20 | 12 | 10 | 23 | 32 | 15 | 11 | 10 | 12 | 11 | 13 | 21 | 23 |
| **ALC-PSO** | 9 | 7 | 17 | 22 | **3** | 4 | 5 | 23 | 14 | 15 | 18 | 5 | 24 | 18 | 9 | 10 | 12 | **3** | **3** | **3** | **2** | **1** |
| **AMALGAM** | 17 | 11 | 17 | 25 | 9 | 7 | **1** | 21 | **3** | 19 | **3** | **3** | 22 | 13 | 23 | 25 | 11 | 7 | 6 | 8 | 6 | 5 |
| **AM-DEGL** | 12 | 21 | 17 | 32 | 20 | 13 | 9 | 11.5 | 27 | 21 | 30 | 27 | **3** | 5 | 17 | 27 | 26 | 28 | 27 | 28 | 19 | 15 |
| **ANS** | 4 | 14 | 17 | 31 | 22 | 18 | 32 | 5 | 15 | 12 | 19 | 16 | 17 | 31 | 12 | 19 | 16 | 14 | 14 | 16 | 17 | 13 |
| **ATPS-DE** | 15 | 17 | 17 | 6 | 13 | 12 | 16 | 5 | 4 | 6 | 5 | 6 | 16 | 27 | **3** | **2** | 6 | 4 | 5 | 4 | 11 | 10 |
| **CDE** | 28 | 20 | 17 | 13 | 18 | 22 | 17 | 22 | 23 | 31 | 14 | 23 | 28 | 22 | 27 | 15 | 24 | 24 | 25 | 25 | 27 | 28 |
| **CLPSO** | 7 | 15 | 17 | 17 | 10 | 17 | 12 | 11.5 | 30 | 26 | 27 | 28 | 15 | 21 | **2** | 30 | 28 | 31 | 31 | 31 | 18 | 22 |
| **CLPSO-DEGL** | **3** | 9 | 17 | 14 | 14 | 10 | 10 | 26 | 8 | 10 | 7 | 13 | 31 | 7 | 6 | **3** | **3** | **2** | **2** | **2** | 7 | **2** |
| **CoBiDE** | 25 | 27 | 17 | 23 | 17 | 15 | 15 | 20 | 21 | 29 | 24 | 22 | 12 | 14 | 28 | 20 | 21 | 23 | 24 | 23 | 26 | 26 |
| **DE** | 23 | 33 | 17 | 30 | 32 | 32 | 27 | 18 | 29 | 32 | 32 | 30 | 25 | 33 | 29 | 29 | 29 | 29 | 29 | 29 | 30 | 31 |
| **DEGL** | 14 | 31 | 17 | 28 | 31 | 30 | 28 | 25 | 24 | **2** | 28 | 26 | 8 | 17 | 18 | 18 | 23 | 22 | 20 | 21 | 24 | 16 |
| **DE-SG** | 19 | 19 | 17 | 26 | 21 | 23 | 19 | 5 | 33 | 28 | 22 | 29 | 6 | 15 | 31 | 31 | 32 | 33 | 33 | 33 | 29 | 30 |
| **EPSDE** | 24 | 25 | 17 | 15 | 23 | 27 | 21 | 11.5 | 9 | 24 | 8 | 12 | 10 | **3** | 8 | 5 | 5 | 8 | 8 | 9 | 15 | 17 |
| **FDSADE** | 5 | 22 | 17 | 24 | 11 | 11 | 20 | 5 | 11 | 16 | 23 | 17 | 9 | 25 | 7 | 14 | 14 | 13 | 13 | 14 | 10 | 11 |
| **GA-MPC** | 20 | 32 | 17 | 18 | 30 | 31 | 33 | 27 | 26 | 11 | 20 | **1** | 5 | 23 | 24 | 22 | 15 | 15 | 17 | 6 | 32 | 25 |
| **GLPSO** | 10 | 10 | 17 | 10 | 12 | 9 | 11 | 14 | 5 | 5 | 13 | 8 | 27 | 30 | 19 | 13 | 8 | 5 | 4 | 5 | 5 | 7 |
| **HCLPSO** | 6 | **3** | 17 | **3** | 4 | **3** | 6 | 5 | 18 | 8 | 4 | 18 | 14 | 24 | 20 | 4 | 4 | 17 | 16 | 17 | **3** | 6 |
| **jDElscop** | **2** | 24 | 17 | 33 | 15 | 21 | 14 | 5 | 22 | 13 | 29 | 24 | **2** | 10 | 16 | 23 | 22 | 25 | 23 | 24 | 20 | 14 |
| **LBBO** | 13 | **1** | 17 | 11 | **1** | **2** | **2** | 5 | 19 | 25 | 17 | 11 | 11 | **2** | 10 | 28 | 27 | 20 | 21 | 19 | 4 | 4 |
| **MDE-pBX** | **1** | 12 | 17 | 9 | 25 | 20 | 25 | 5 | 6 | **1** | 6 | 9 | 18 | 29 | 11 | 21 | 7 | 6 | 7 | 7 | 9 | 8 |
| **MPEDE** | 16 | 30 | 17 | 5 | 29 | 29 | 24 | 17 | 25 | 7 | 25 | 25 | 7 | 6 | 5 | 16 | 25 | 26 | 26 | 26 | 23 | 24 |
| **NMA** | 31 | **2** | 17 | 19 | 8 | 5 | **3** | 31 | 32 | 9 | 26 | 31 | 32 | 4 | 21 | 32 | 30 | 30 | 30 | 30 | 8 | 12 |
| **PMS** | 21 | 4 | 17 | 7 | **2** | **1** | 8 | 33 | **1** | 27 | **1** | 4 | 29 | 8 | 26 | 17 | 18 | 18 | 18 | 18 | 13 | 18 |
| **PSO-iw** | 8 | 6 | 17 | **1** | 5 | 6 | 4 | 30 | 7 | 4 | 10 | 7 | 21 | 16 | 14 | 8 | **2** | **1** | **1** | **1** | **1** | **3** |
| **RA** | 33 | 5 | 17 | 29 | 6 | 8 | 18 | 32 | **2** | 33 | **2** | **2** | 33 | 9 | 32 | 12 | 17 | 19 | 19 | 20 | 33 | 33 |
| **Rcr-JADE** | 30 | 28 | 17 | **2** | 28 | 26 | 31 | 19 | 12 | 18 | 11 | 14 | 20 | 28 | 25 | 7 | 9 | 10 | 10 | 11 | 25 | 27 |
| **SADE** | 26 | 18 | 17 | 16 | 24 | 16 | 23 | 15 | 13 | 14 | 9 | 15 | 4 | **1** | **1** | **1** | **1** | 11 | 12 | 12 | 22 | 20 |
| **SapsDE** | 18 | 16 | 17 | 20 | 16 | 14 | 22 | 11.5 | 16 | **3** | 16 | 20 | 13 | 26 | 4 | 6 | 13 | 9 | 9 | 10 | 12 | 9 |
| **SFMDE** | 22 | 8 | 17 | 8 | 7 | 28 | 7 | 28 | 17 | 22 | 21 | 19 | 19 | 11 | 13 | 24 | 20 | 16 | 15 | 15 | 14 | 21 |
| **SPS-L-SHADEEIG** | 11 | 23 | 17 | 21 | 19 | 19 | 13 | 5 | 28 | 17 | 31 | 33 | **1** | 12 | 22 | 26 | 33 | 27 | 28 | 27 | 16 | 19 |
| **SP-UCI** | 32 | 13 | 17 | 27 | 33 | 33 | 30 | 29 | 31 | 23 | 33 | 32 | 30 | 20 | 33 | 33 | 31 | 32 | 32 | 32 | 31 | 32 |
| **SspDE** | 29 | 26 | 17 | 4 | 27 | 24 | 26 | 24 | 20 | 30 | 15 | 21 | 26 | 19 | 30 | 9 | 19 | 21 | 22 | 22 | 28 | 29 |

**Table 4**
Ranking of algorithms on all problems with allowed number of function calls set to 20,000. Algorithms with ranks 1–3 are bolded.

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11.1 | P11.2 | P11.3 | P11.4 | P11.5 | P11.6 | P11.7 | P11.8 | P11.9 | P11.10 | P12 | P13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AdapSS-JADE** | 19 | 20 | 17 | 15 | 21 | 19 | 25 | 13 | 10 | 17 | 6 | 6 | 17 | 23 | 22 | 15 | 12 | 16 | 15 | 15 | 17 | 22 |
| **ALC-PSO** | 28 | 8 | 17 | 5 | 12 | 8 | 6 | 13 | 23 | 29 | 20 | 10 | 29 | 28 | 24 | 30 | 10 | 14 | 11 | 13 | 7 | 12 |
| **AMALGAM** | 26 | 17 | 17 | 22 | 20 | 18 | **1** | 26 | 5 | 26 | **2** | **3** | 27 | 21 | 17 | 27 | 18 | 12 | 9 | 12 | 9 | 10 |
| **AM-DEGL** | 8 | 10 | 17 | 31 | 13 | 21 | 7 | 13 | 29 | 13 | 30 | 27 | 8 | **3** | 8 | 19 | 26 | 27 | 26 | 26 | 14 | 17 |
| **ANS** | 12 | 28 | 17 | 25 | 29 | 29 | 32 | 13 | 17 | 21 | 17 | 15 | 25 | 32 | 28 | 25 | 17 | 18 | 18 | 17 | 28 | 26 |
| **ATPS-DE** | 9 | 11 | 17 | 10 | 4 | **3** | 10 | 13 | **2** | 6 | 4 | 5 | 15 | 16 | 13 | 11 | 4 | 5 | 5 | 5 | 10 | 8 |
| **CDE** | 25 | 26 | 17 | 12 | 24 | 24 | 24 | 13 | 19 | 22 | 21 | 21 | 5 | 19 | 15 | 18 | 20 | 20 | 20 | 19 | 26 | 29 |
| **CLPSO** | 10 | 18 | 17 | 11 | 11 | 15 | 14 | 13 | 28 | 27 | 29 | 28 | 20 | 30 | 20 | 13 | 29 | 28 | 29 | 29 | 16 | 21 |
| **CLPSO-DEGL** | 24 | 12 | 17 | 8 | 17 | 10 | 15 | 27 | 12 | 25 | 14 | 16 | 31 | 22 | 29 | 14 | 8 | 8 | 8 | 8 | 6 | 9 |
| **CoBiDE** | 23 | 30 | 17 | 26 | 25 | 25 | 19 | 13 | 21 | 24 | 22 | 22 | 4 | 14 | 11 | 9 | 16 | 19 | 19 | 20 | 24 | 23 |
| **DE** | 17 | 33 | 17 | 33 | 32 | 32 | 31 | 13 | 30 | 30 | 33 | 30 | 22 | 33 | 25 | 29 | 30 | 30 | 30 | 30 | 32 | 32 |
| **DEGL** | 13 | 31 | 17 | 27 | 31 | 30 | 29 | 13 | 22 | 16 | 28 | 25 | 18 | 9 | **3** | 20 | 19 | 17 | 17 | 18 | 27 | 6 |
| **DE-SG** | 6 | 19 | 17 | 30 | 26 | 23 | 20 | 13 | 31 | 20 | 27 | 31 | 12 | 15 | 18 | 10 | 31 | 32 | 32 | 32 | 29 | 28 |
| **EPSDE** | 18 | 29 | 17 | 20 | 27 | 28 | 27 | 13 | 6 | 10 | 10 | 9 | **3** | 7 | 6 | **2** | **1** | 4 | **2** | **3** | 20 | 20 |
| **FDSADE** | 11 | 14 | 17 | 29 | 8 | 6 | 12 | 13 | 13 | 14 | 18 | 12 | 16 | 26 | 16 | 16 | 14 | 15 | 16 | 16 | 5 | **2** |
| **GA-MPC** | **2** | 32 | 17 | 16 | 30 | 31 | 33 | 13 | 25 | 5 | 19 | **1** | **1** | 20 | 10 | 26 | 23 | 11 | 13 | 14 | 25 | **1** |
| **GLPSO** | 15 | **3** | 17 | 13 | **3** | 5 | **3** | 13 | 8 | 19 | 12 | 11 | 28 | 31 | 26 | 24 | 11 | 13 | 12 | 11 | **2** | 4 |
| **HCLPSO** | 16 | **2** | 17 | **1** | 5 | 9 | 8 | 13 | 15 | 11 | 11 | 17 | 21 | 29 | 19 | 12 | 13 | 10 | 14 | 9 | **3** | 11 |
| **jDElscop** | 4 | 21 | 17 | 32 | 16 | 16 | 18 | 13 | 20 | 7 | 31 | 23 | 9 | 12 | 14 | 21 | 22 | 21 | 21 | 21 | 22 | 7 |
| **LBBO** | 22 | **1** | 17 | 6 | **1** | **2** | 4 | 13 | 16 | 31 | **3** | 14 | 19 | 13 | 21 | 33 | 24 | 26 | 25 | 25 | 8 | 13 |
| **MDE-pBX** | **1** | 5 | 17 | 17 | 14 | 13 | 28 | 13 | **3** | 4 | 7 | 7 | 14 | 10 | 4 | 8 | 9 | **1** | **1** | **2** | **1** | **3** |
| **MPEDE** | 7 | 27 | 17 | 14 | 28 | 27 | 22 | 13 | 24 | 12 | 23 | 24 | 11 | **2** | **2** | 4 | 21 | 22 | 22 | 22 | 21 | 18 |
| **NMA** | 31 | 4 | 17 | 7 | 6 | 11 | **2** | 31 | 33 | 23 | 32 | 33 | 30 | 17 | 27 | 22 | 33 | 33 | 33 | 33 | 12 | 24 |
| **PMS** | 29 | 6 | 17 | **3** | **2** | **1** | 13 | 33 | **1** | 32 | **1** | **2** | 32 | 25 | 32 | 32 | 15 | 29 | 28 | 28 | 30 | 30 |
| **PSO-iw** | 21 | 7 | 17 | **2** | 10 | 14 | 5 | 30 | 18 | 18 | 16 | 19 | 26 | 27 | 31 | 23 | 6 | 9 | 10 | 10 | 4 | 15 |
| **RA** | 33 | 13 | 17 | 18 | 9 | 20 | 21 | 32 | 4 | 33 | 8 | 4 | 33 | 18 | 33 | 28 | 25 | 25 | 27 | 27 | 33 | 33 |
| **Rcr-JADE** | 20 | 22 | 17 | 4 | 23 | 26 | 26 | 13 | 7 | **1** | 5 | 8 | 7 | 6 | **1** | 5 | 7 | 7 | 4 | 6 | 19 | 19 |
| **SADE** | 14 | 15 | 17 | 24 | 19 | 12 | 17 | 13 | 9 | **3** | 9 | 13 | 10 | 8 | 5 | **3** | **3** | 6 | 6 | 7 | 13 | 14 |
| **SapsDE** | **3** | 9 | 17 | 21 | 7 | 4 | 16 | 13 | 11 | 9 | 13 | 18 | 13 | 11 | 12 | 7 | 5 | **2** | **3** | 4 | 11 | 5 |
| **SFMDE** | 30 | 16 | 17 | 9 | 15 | 7 | 9 | 28 | 26 | 28 | 24 | 26 | 23 | 24 | 30 | 17 | 27 | 24 | 23 | 23 | 18 | 27 |
| **SPS-L-SHADEEIG** | 5 | 23 | 17 | 23 | 18 | 17 | 11 | 13 | 27 | 8 | 26 | 29 | **2** | 5 | 7 | 6 | 28 | 23 | 24 | 24 | 15 | 16 |
| **SP-UCI** | 32 | 25 | 17 | 28 | 33 | 33 | 30 | 29 | 32 | **2** | 25 | 32 | 24 | **1** | 23 | 31 | 32 | 31 | 31 | 31 | 31 | 31 |
| **SspDE** | 27 | 24 | 17 | 19 | 22 | 22 | 23 | 13 | 14 | 15 | 15 | 20 | 6 | 4 | 9 | **1** | **2** | **3** | 7 | **1** | 23 | 25 |

**Table 5**
Ranking of algorithms on all problems with allowed number of function calls set to 50,000. Algorithms with ranks 1–3 are bolded.

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11.1 | P11.2 | P11.3 | P11.4 | P11.5 | P11.6 | P11.7 | P11.8 | P11.9 | P11.10 | P12 | P13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AdapSS-JADE** | 16 | 17 | 17 | 13 | 12 | 5 | 19 | 14 | **2** | 7 | **1** | 4 | 8 | 17 | 18 | 13 | 9 | 7 | 7 | 8 | 21 | 17 |
| **ALC-PSO** | 28 | 13 | 17 | **2** | 22 | 20 | 12 | 14 | 29 | 30 | 27 | 15 | 30 | 30 | 29 | 29 | 16 | 17 | 16 | 19 | 10 | 20 |
| **AMALGAM** | 26 | 16 | 17 | 18 | 26 | 27 | **1** | 14 | 11 | 26 | 6 | **3** | 28 | 10 | 13 | 27 | 20 | 10 | **3** | 10 | 14 | 13 |
| **AM-DEGL** | 15 | 9 | 17 | 29 | 21 | 23 | 8 | 14 | 26 | 9 | 20 | 26 | 7 | 7 | 6 | 5 | 18 | 23 | 20 | 22 | 15 | 12 |
| **ANS** | **2** | 30 | 17 | 19 | 30 | 30 | 33 | 14 | 19 | 29 | 17 | 17 | 26 | 32 | 28 | 25 | 19 | 22 | 23 | 23 | 27 | 29 |
| **ATPS-DE** | 12 | 7 | 17 | 16 | **2** | **2** | 9 | 14 | 7 | 10 | 8 | **2** | 16 | 20 | 19 | 14 | 8 | 6 | 4 | 6 | 6 | 14 |
| **CDE** | 5 | 26 | 17 | 24 | 25 | 22 | 26 | 14 | 16 | 5 | 19 | 18 | 12 | 14 | 11 | 21 | 13 | 19 | 19 | 18 | 28 | 25 |
| **CLPSO** | 17 | 23 | 17 | 12 | 7 | 10 | 16 | 14 | 25 | 23 | 28 | 27 | 23 | 27 | 23 | 20 | 25 | 25 | 25 | 25 | 17 | 22 |
| **CLPSO-DEGL** | 25 | 11 | 17 | 9 | 15 | 16 | 13 | 29 | 13 | 25 | 13 | 16 | 31 | 24 | 30 | 17 | 10 | 14 | 10 | 15 | **3** | 18 |
| **CoBiDE** | 8 | 28 | 17 | 26 | 23 | 21 | 21 | 14 | 18 | 11 | 25 | 21 | 4 | 11 | 9 | 6 | 6 | 9 | 8 | 7 | 25 | **2** |
| **DE** | 13 | 33 | 17 | 33 | 32 | 32 | 31 | 14 | 30 | 28 | 33 | 31 | 20 | 33 | 26 | 28 | 31 | 31 | 31 | 31 | 32 | 32 |
| **DEGL** | 20 | 32 | 17 | 32 | 31 | 31 | 30 | 14 | 17 | 15 | 24 | 25 | 21 | 6 | **3** | 19 | 22 | 21 | 21 | 20 | 29 | 5 |
| **DE-SG** | 6 | 24 | 17 | 28 | 24 | 18 | 25 | 14 | 31 | 18 | 31 | 32 | 14 | 18 | 20 | 11 | 32 | 32 | 32 | 32 | 26 | 24 |
| **EPSDE** | 14 | 29 | 17 | 23 | 28 | 28 | 29 | 14 | 9 | **2** | 10 | 8 | 10 | 8 | 8 | 9 | **3** | 5 | 6 | 5 | 24 | 11 |
| **FDSADE** | 21 | 6 | 17 | 27 | 14 | 13 | 5 | 14 | 10 | 19 | 15 | 9 | 18 | 23 | 17 | 16 | 14 | 15 | 15 | 12 | 8 | **3** |
| **GA-MPC** | **3** | 27 | 17 | 11 | 29 | 29 | 27 | 14 | 24 | **1** | 18 | **1** | 6 | 19 | 10 | 24 | 27 | 16 | 14 | 16 | **1** | **1** |
| **GLPSO** | 23 | **3** | 17 | 20 | 8 | 11 | **3** | 14 | 6 | 21 | 7 | 12 | 25 | 31 | 25 | 26 | 15 | 13 | 11 | 14 | **2** | 19 |
| **HCLPSO** | 19 | **1** | 17 | 4 | 5 | 14 | 10 | 14 | 12 | 20 | 4 | 14 | 19 | 28 | 22 | 18 | 17 | 11 | 17 | 13 | 5 | 23 |
| **jDElscop** | **1** | 21 | 17 | 31 | 17 | 12 | 22 | 14 | 20 | 6 | 30 | 20 | 5 | 15 | 14 | 22 | 24 | 20 | 22 | 21 | 22 | 6 |
| **LBBO** | 24 | **2** | 17 | **1** | **1** | 6 | 4 | 14 | 21 | 32 | 5 | 22 | 22 | 16 | 21 | 33 | 29 | 29 | 27 | 28 | 16 | 21 |
| **MDE-pBX** | 4 | 5 | 17 | 15 | 18 | 17 | 28 | 14 | 4 | 4 | 11 | 7 | 15 | 4 | 4 | 10 | 12 | **2** | **2** | **2** | 4 | 9 |
| **MPEDE** | 9 | 15 | 17 | 10 | 19 | 15 | 15 | 14 | 22 | 17 | 22 | 23 | 13 | 5 | 5 | **3** | 7 | 18 | 18 | 17 | 11 | 7 |
| **NMA** | 32 | 4 | 17 | 6 | 6 | 24 | **2** | 28 | 33 | 22 | 32 | 33 | 32 | 21 | 24 | 7 | 33 | 33 | 33 | 33 | 19 | 30 |
| **PMS** | 30 | 10 | 17 | **3** | **3** | **3** | 18 | 33 | **1** | 31 | **3** | 6 | 33 | 26 | 33 | 32 | 21 | 30 | 30 | 30 | 30 | 31 |
| **PSO-iw** | 27 | 12 | 17 | 5 | 20 | 25 | 6 | 31 | 23 | 24 | 21 | 24 | 29 | 29 | 31 | 30 | 11 | 12 | 13 | 11 | 7 | 26 |
| **RA** | 33 | 20 | 17 | 7 | 9 | 26 | 24 | 32 | 14 | 33 | 16 | 10 | 24 | 22 | 32 | 31 | 28 | 28 | 29 | 27 | 33 | 33 |
| **Rcr-JADE** | 7 | 18 | 17 | 8 | 16 | 7 | 20 | 14 | **3** | **3** | **2** | 5 | **3** | **3** | **2** | **1** | **1** | **1** | **1** | **1** | 20 | 16 |
| **SADE** | 10 | 14 | 17 | 21 | 11 | 4 | 17 | 14 | 8 | 12 | 12 | 11 | 9 | 13 | 16 | 12 | 4 | 8 | 12 | 9 | 13 | 8 |
| **SapsDE** | 18 | 8 | 17 | 22 | 4 | **1** | 14 | 14 | 5 | 16 | 9 | 13 | 11 | 12 | 15 | 15 | 5 | **3** | 5 | **3** | 9 | 10 |
| **SFMDE** | 29 | 25 | 17 | 17 | 27 | 19 | 11 | 14 | 28 | 27 | 29 | 28 | 17 | 25 | 27 | 23 | 26 | 27 | 28 | 29 | 18 | 27 |
| **SPS-L-SHADEEIG** | 11 | 22 | 17 | 25 | 10 | 9 | 7 | 14 | 27 | 8 | 26 | 29 | **1** | **2** | 7 | **2** | 23 | 24 | 24 | 24 | 12 | 4 |
| **SP-UCI** | 31 | 31 | 17 | 30 | 33 | 33 | 32 | 30 | 32 | 14 | 23 | 30 | 27 | **1** | **1** | 4 | 30 | 26 | 26 | 26 | 31 | 28 |
| **SspDE** | 22 | 19 | 17 | 14 | 13 | 8 | 23 | 14 | 15 | 13 | 14 | 19 | **2** | 9 | 12 | 8 | **2** | 4 | 9 | 4 | 23 | 15 |

**Table 6**
Ranking of algorithms on all problems with allowed number of function calls set to 100,000. Algorithms with ranks 1–3 are bolded.

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11.1 | P11.2 | P11.3 | P11.4 | P11.5 | P11.6 | P11.7 | P11.8 | P11.9 | P11.10 | P12 | P13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AdapSS-JADE** | 10 | 17 | 17 | 20 | 13 | 4 | 23 | 14.5 | 5 | 13 | **3** | **3** | 9 | 15 | 20 | 12 | 7 | 5 | 9 | 6 | 19 | 13 |
| **ALC-PSO** | 29 | 22 | 17 | **3** | 25 | 23 | 13 | 14.5 | 29 | 30 | 30 | 22 | 31 | 30 | 23 | 29 | 21 | 23 | 21 | 24 | 23 | 27 |
| **AMALGAM** | 24 | 23 | 17 | 13 | 27 | 26 | **1** | 14.5 | 15 | 25 | 11 | 5 | 26 | **2** | 10 | 25 | 25 | 10 | 4 | 11 | 13 | 20 |
| **AM-DEGL** | 14 | 5 | 17 | 29 | 8 | 17 | 7 | 14.5 | 23 | 7 | 20 | 25 | 17 | 9 | **3** | **3** | 11 | 17 | 8 | 9 | 10 | 7 |
| **ANS** | 9 | 30 | 17 | 15 | 30 | 31 | 32 | 14.5 | 22 | 28 | 23 | 21 | 27 | 32 | 29 | 28 | 23 | 26 | 26 | 26 | 28 | 26 |
| **ATPS-DE** | 17 | 8 | 17 | 12 | **3** | **2** | 10 | 14.5 | 11 | 14 | 9 | 4 | 14 | 19 | 16 | 10 | 8 | 9 | 5 | 7 | 6 | 16 |
| **CDE** | **1.5** | 25 | 17 | 22 | 26 | 20 | 28 | 14.5 | 17 | **2** | 14 | 14 | 11 | 6 | 9 | 17 | 13 | 20 | 20 | 20 | 30 | 17 |
| **CLPSO** | 11 | 20 | 17 | 10 | 7 | 12 | 19 | 14.5 | 24 | 21 | 27 | 24 | 22 | 28 | 27 | 18 | 24 | 24 | 22 | 23 | 12 | 19 |
| **CLPSO-DEGL** | 26 | 12 | 17 | 16 | 16 | 19 | 9 | 31 | 16 | 27 | 18 | 16 | 32 | 25 | 30 | 20 | 16 | 16 | 13 | 16 | **3** | 25 |
| **CoBiDE** | 7 | 26 | 17 | 25 | 11 | 13 | **2** | 14.5 | 7 | **1** | 19 | 18 | 4 | **3** | 6 | 6 | **3** | **3** | **3** | **2** | 11 | **2** |
| **DE** | 6 | 33 | 17 | 33 | 32 | 32 | 31 | 14.5 | 30 | 26 | 33 | 31 | 12 | 33 | 22 | 27 | 32 | 31 | 32 | 32 | 32 | 31 |
| **DEGL** | 23 | 31 | 17 | 31 | 31 | 30 | 30 | 29.5 | 19 | 19 | 22 | 23 | 23 | 10 | 5 | 16 | 19 | 25 | 25 | 25 | 17 | 11 |
| **DE-SG** | 16 | 24 | 17 | 30 | 23 | 15 | 27 | 14.5 | 32 | 11 | 32 | 32 | 13 | 20 | 21 | 13 | 31 | 32 | 31 | 31 | 27 | 18 |
| **EPSDE** | 5 | 29 | 17 | 24 | 28 | 29 | 29 | 14.5 | 8 | 9 | 5 | 7 | 5 | 13 | 11 | 9 | 4 | 8 | 10 | 8 | 24 | 10 |
| **FDSADE** | 22 | 9 | 17 | 26 | 21 | 21 | 6 | 14.5 | 9 | 17 | 12 | 6 | 20 | 23 | 15 | 21 | 15 | 13 | 16 | 13 | **2** | 5 |
| **GA-MPC** | 13 | 4 | 17 | 8 | 18 | 18 | 14 | 14.5 | 20 | **3** | 15 | **1** | **1** | 18 | 12 | 24 | 17 | 21 | 15 | 21 | **1** | **1** |
| **GLPSO** | 25 | **1** | 17 | 19 | 20 | 16 | 8 | 14.5 | 14 | 22 | 6 | 15 | 30 | 31 | 28 | 26 | 18 | 14 | 17 | 10 | 8 | 23 |
| **HCLPSO** | 19 | **3** | 17 | 6 | 5 | 11 | 11 | 14.5 | **2** | 16 | 10 | 12 | 18 | 27 | 24 | 19 | 22 | 18 | 24 | 18 | 9 | 21 |
| **jDElscop** | **1.5** | 21 | 17 | 32 | 12 | 9 | 21 | 14.5 | 21 | 8 | 29 | 19 | 7 | 16 | 13 | 22 | 26 | 22 | 23 | 22 | 21 | 4 |
| **LBBO** | 21 | 7 | 17 | **1** | **1** | 14 | 4 | 14.5 | 25 | 31 | **1** | 26 | 21 | 17 | 19 | 33 | 30 | 27 | 27 | 28 | 15 | 22 |
| **MDE-pBX** | 18 | 6 | 17 | 18 | 22 | 22 | 22 | 14.5 | 12 | 15 | 4 | 8 | 16 | 5 | 8 | 8 | 14 | 4 | **2** | 4 | 7 | 14 |
| **MPEDE** | 20 | 11 | 17 | 9 | 10 | 10 | 15 | 14.5 | 18 | 18 | 17 | 17 | 19 | 8 | 7 | 4 | **2** | **2** | 6 | **1** | 4 | 8 |
| **NMA** | 32 | **2** | 17 | 7 | 4 | 28 | **3** | 29.5 | 33 | 24 | 31 | 33 | 29 | 21 | 26 | 7 | 33 | 33 | 33 | 33 | 29 | 30 |
| **PMS** | 27 | 19 | 17 | 4 | 17 | 7 | 26 | 33 | **1** | 32 | **2** | 11 | 33 | 26 | 33 | 32 | 27 | 30 | 30 | 30 | 31 | 32 |
| **PSO-iw** | 28 | 14 | 17 | 5 | 24 | 27 | 12 | 14.5 | 27 | 23 | 26 | 27 | 25 | 29 | 31 | 30 | 12 | 12 | 14 | 17 | 20 | 28 |
| **RA** | 33 | 27 | 17 | **2** | 19 | 24 | 25 | 32 | 4 | 33 | 21 | 13 | 24 | 22 | 32 | 31 | 29 | 28 | 28 | 27 | 33 | 33 |
| **Rcr-JADE** | **3** | 16 | 17 | 11 | 15 | 6 | 20 | 14.5 | 6 | 5 | 8 | **2** | 8 | 7 | **2** | 5 | **1** | **1** | **1** | **3** | 18 | 12 |
| **SADE** | 4 | 13 | 17 | 27 | 14 | 5 | 17 | 14.5 | **3** | 6 | 13 | 9 | 10 | 14 | 18 | 14 | 6 | 11 | 12 | 14 | 16 | 6 |
| **SapsDE** | 12 | 10 | 17 | 21 | **2** | **1** | 18 | 14.5 | 10 | 20 | 7 | 10 | 15 | 11 | 14 | 15 | 10 | 6 | 7 | 5 | 14 | 15 |
| **SFMDE** | 31 | 28 | 17 | 14 | 29 | 25 | 16 | 14.5 | 28 | 29 | 28 | 29 | 6 | 24 | 25 | 23 | 28 | 29 | 29 | 29 | 25 | 29 |
| **SPS-L-SHADEEIG** | 8 | 15 | 17 | 23 | 6 | 8 | 5 | 14.5 | 26 | 4 | 25 | 28 | **2** | **1** | 4 | **2** | 9 | 15 | 18 | 15 | 5 | **3** |
| **SP-UCI** | 30 | 32 | 17 | 28 | 33 | 33 | 33 | 14.5 | 31 | 10 | 24 | 30 | 28 | 4 | **1** | **1** | 20 | 19 | 19 | 19 | 26 | 24 |
| **SspDE** | 15 | 18 | 17 | 17 | 9 | **3** | 24 | 14.5 | 13 | 12 | 16 | 20 | **3** | 12 | 17 | 11 | 5 | 7 | 11 | 12 | 22 | 9 |

**Table 7**
Ranking of algorithms on all problems with allowed number of function calls set to 150,000. Algorithms with ranks 1–3 are bolded.

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11.1 | P11.2 | P11.3 | P11.4 | P11.5 | P11.6 | P11.7 | P11.8 | P11.9 | P11.10 | P12 | P13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AdapSS-JADE** | 9 | 15 | 17 | 20 | 7 | **3** | 23 | 14.5 | 9 | 14 | 4 | **3** | 8 | 17 | 22 | 14 | 8 | 5 | 10 | 5 | 22 | 16 |
| **ALC-PSO** | 26 | 26 | 17 | **3** | 24 | 23 | 18 | 14.5 | 30 | 30 | 31 | 24 | 29 | 31 | 21 | 29 | 20 | 24 | 23 | 24 | 26 | 27 |
| **AMALGAM** | 22 | 22 | 17 | 10 | 28 | 27 | **1** | 14.5 | 18 | 25 | **3** | 4 | 27 | **2** | 8 | 21 | 26 | 14 | 5 | 15 | 15 | 21 |
| **AM-DEGL** | 11 | 6 | 17 | 31 | 11 | 14 | 7 | 14.5 | 20 | 7 | 19 | 25 | 17 | 9 | 5 | **3** | 12 | 6 | 7 | 7 | 11 | 6 |
| **ANS** | **3** | 30 | 17 | 11 | 30 | 31 | 32 | 14.5 | 24 | 28 | 25 | 21 | 25 | 32 | 27 | 26 | 22 | 26 | 27 | 27 | 27 | 29 |
| **ATPS-DE** | 18 | 8 | 17 | 13 | 5 | 6 | 10 | 14.5 | 15 | 17 | 11 | 6 | 13 | 20 | 17 | 10 | 5 | 9 | 8 | 8 | 5 | 17 |
| **CDE** | **3** | 25 | 17 | 22 | 26 | 17 | 28 | 14.5 | 14 | **3** | **2** | 10 | 10 | 8 | 9 | 16 | 13 | 19 | 22 | 19 | 29 | 12 |
| **CLPSO** | 15 | 21 | 17 | 14 | 10 | 9 | 17 | 14.5 | 23 | 21 | 26 | 23 | 21 | 28 | 23 | 23 | 23 | 21 | 19 | 20 | 9 | 18 |
| **CLPSO-DEGL** | 29 | 9 | 17 | 17 | 16 | 19 | 13 | 30 | 19 | 24 | 20 | 15 | 32 | 23 | 30 | 20 | 17 | 16 | 11 | 16 | 6 | 24 |
| **CoBiDE** | **3** | 19 | 17 | 28 | 14 | 10 | **2** | 14.5 | 4 | **1** | 5 | 16 | **1** | 4 | **2** | 6 | **3** | **3** | **3** | **3** | **3** | 4 |
| **DE** | **3** | 33 | 17 | 29 | 32 | 32 | 31 | 14.5 | 31 | 22 | 33 | 31 | 9 | 33 | 24 | 28 | 33 | 33 | 33 | 33 | 32 | 30 |
| **DEGL** | 24 | 31 | 17 | 32 | 31 | 30 | 30 | 14.5 | 21 | 26 | 22 | 22 | 22 | 10 | 7 | 15 | 19 | 25 | 25 | 25 | 10 | 13 |
| **DE-SG** | 10 | 24 | 17 | 30 | 22 | 15 | 26 | 14.5 | 32 | 10 | 32 | 32 | 18 | 16 | 20 | 12 | 29 | 30 | 30 | 30 | 28 | 11 |
| **EPSDE** | 8 | 27 | 17 | 23 | 27 | 29 | 29 | 14.5 | 12 | 8 | 13 | 8 | 12 | 13 | 13 | 11 | 6 | 8 | 13 | 9 | 25 | 5 |
| **FDSADE** | 23 | 13 | 17 | 27 | 21 | 20 | 6 | 14.5 | 13 | 15 | 14 | 5 | 23 | 24 | 14 | 22 | 15 | 13 | 20 | 17 | 21 | 9 |
| **GA-MPC** | 16 | **1** | 17 | 8 | 17 | 18 | 9 | 14.5 | 7 | 4 | 16 | **1** | 5 | 18 | 11 | 25 | 24 | 20 | 17 | 22 | **1** | **1** |
| **GLPSO** | 25 | **3** | 17 | 16 | 20 | 21 | 12 | 14.5 | 17 | 20 | 6 | 19 | 31 | 30 | 28 | 27 | 21 | 12 | 18 | 12 | 13 | 25 |
| **HCLPSO** | 13 | 5 | 17 | 6 | 3 | 11 | 16 | 14.5 | **3** | 16 | 17 | 12 | 15 | 27 | 25 | 24 | 18 | 17 | 21 | 13 | 8 | 22 |
| **jDElscop** | **3** | 20 | 17 | 33 | 15 | 7 | 20 | 14.5 | 22 | 6 | 27 | 18 | 11 | 11 | 12 | 19 | 25 | 23 | 24 | 23 | 23 | **2** |
| **LBBO** | 17 | 7 | 17 | **2** | **2** | 16 | 5 | 14.5 | 26 | 31 | **1** | 27 | 20 | 19 | 19 | 33 | 32 | 29 | 29 | 28 | 17 | 20 |
| **MDE-pBX** | 20 | 12 | 17 | 19 | 23 | 22 | 22 | 14.5 | 10 | 9 | 12 | 7 | 14 | **3** | 10 | 8 | 4 | 4 | 4 | 4 | 12 | 19 |
| **MPEDE** | 21 | **2** | 17 | 12 | 12 | 12 | 8 | 14.5 | 16 | 19 | 8 | 14 | 16 | 7 | 6 | 4 | **1** | **1** | **1** | **1** | 4 | 7 |
| **NMA** | 33 | 4 | 17 | 5 | 13 | 26 | 4 | 14.5 | 33 | 27 | 28 | 33 | 30 | 22 | 29 | 7 | 31 | 28 | 28 | 29 | 30 | 31 |
| **PMS** | 30 | 23 | 17 | 4 | 18 | 13 | 27 | 33 | **2** | 32 | 7 | 13 | 33 | 26 | 33 | 32 | 28 | 32 | 32 | 32 | 31 | 32 |
| **PSO-iw** | 27 | 17 | 17 | 7 | 25 | 28 | 14 | 31 | 28 | 29 | 30 | 28 | 26 | 29 | 32 | 30 | 16 | 22 | 15 | 21 | 16 | 26 |
| **RA** | 32 | 29 | 17 | **1** | 19 | 24 | 25 | 32 | **1** | 33 | 24 | 17 | 24 | 21 | 31 | 31 | 27 | 27 | 26 | 26 | 33 | 33 |
| **Rcr-JADE** | 7 | 16 | 17 | 9 | 4 | 5 | 21 | 14.5 | 8 | 13 | 10 | **2** | **3** | 6 | **3** | 5 | **2** | **2** | **2** | **2** | 20 | 14 |
| **SADE** | 6 | 14 | 17 | 21 | 6 | 4 | 15 | 14.5 | 6 | 5 | 9 | 11 | 7 | 15 | 16 | 13 | 10 | 18 | 16 | 18 | 18 | 15 |
| **SapsDE** | 19 | 10 | 17 | 24 | **1** | **1** | 19 | 14.5 | 11 | 18 | 18 | 9 | 19 | 12 | 18 | 17 | 11 | 11 | 9 | 6 | 14 | 8 |
| **SFMDE** | 31 | 28 | 17 | 18 | 29 | 25 | 11 | 14.5 | 27 | 23 | 29 | 30 | 6 | 25 | 26 | 18 | 30 | 31 | 31 | 31 | 24 | 28 |
| **SPS-L-SHADEEIG** | 14 | 11 | 17 | 25 | 9 | 8 | **3** | 14.5 | 25 | **2** | 23 | 26 | **2** | **1** | **1** | **1** | 9 | 10 | 12 | 11 | **2** | **3** |
| **SP-UCI** | 28 | 32 | 17 | 26 | 33 | 33 | 33 | 29 | 29 | 12 | 21 | 29 | 28 | 5 | 4 | **2** | 14 | 15 | 6 | 14 | 7 | 23 |
| **SspDE** | 12 | 18 | 17 | 15 | 8 | **2** | 24 | 14.5 | 5 | 11 | 15 | 20 | 4 | 14 | 15 | 9 | 7 | 7 | 14 | 10 | 19 | 10 |

**Table 8**
Ranking of algorithms on all problems with allowed number of function calls set to 500,000. Algorithms with ranks 1–3 are bolded.

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11.1 | P11.2 | P11.3 | P11.4 | P11.5 | P11.6 | P11.7 | P11.8 | P11.9 | P11.10 | P12 | P13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AdapSS-JADE** | 11 | 13 | 17 | 19 | **1** | 5 | 25 | 15.5 | 16 | 11 | 12 | 6 | 12 | 22 | 27 | 17 | 10 | 7 | 12 | 6 | 22 | 14 |
| **ALC-PSO** | 25 | 26 | 17 | **2** | 23 | 22 | 23 | 15.5 | 32 | 31 | 32 | 27 | 26 | 32 | 20 | 24 | 29 | 27 | 26 | 27 | 27 | 29 |
| **AMALGAM** | 9 | 19 | 17 | 11 | 27 | 25 | **2** | 15.5 | 21 | 24 | 7 | **2** | 25 | **2** | 10 | 15 | 27 | 20 | 8 | 18 | 12 | 19 |
| **AM-DEGL** | 20 | 6 | 17 | 32 | 11 | **1** | 5 | 15.5 | **2** | 5 | **3** | 13 | 19 | 5 | 6 | **3** | **3** | **3** | **3** | **3** | 13 | 7 |
| **ANS** | 15 | 30 | 17 | 9 | 31 | 31 | 31 | 15.5 | 26 | 29 | 29 | 25 | 28 | 31 | 23 | 28 | 28 | 29 | 30 | 29 | 29 | 28 |
| **ATPS-DE** | 18 | 9 | 17 | 16 | 8 | 11 | 10 | 15.5 | 20 | 15 | 20 | 10 | 15 | 19 | 16 | 12 | 8 | 10 | 10 | 12 | 8 | 17 |
| **CDE** | 4 | 23 | 17 | 20 | 22 | 13 | 28 | 15.5 | **1** | 12 | **2** | 9 | 9 | 9 | 8 | 21 | 17 | 15 | 25 | 15 | 18 | **3** |
| **CLPSO** | 16 | 18 | 17 | 13 | 10 | 7 | 18 | 15.5 | 11 | 20 | 24 | 14 | 21 | 29 | 22 | 26 | 18 | 21 | 19 | 20 | **3** | 13 |
| **CLPSO-DEGL** | 30 | 11 | 17 | 12 | 15 | 24 | 13 | 15.5 | 24 | 28 | 21 | 20 | 33 | 24 | 29 | 22 | 20 | 24 | 13 | 22 | 11 | 24 |
| **CoBiDE** | 4 | **2** | 17 | 23 | 14 | 15 | **3** | 15.5 | 8 | **3** | 8 | 17 | **1** | 4 | **2** | 7 | 6 | 8 | 5 | 7 | 5 | 4 |
| **DE** | 4 | 33 | 17 | 30 | 32 | 32 | 32 | 15.5 | 33 | 18 | 33 | 32 | 6 | 33 | 24 | 27 | 31 | 33 | 33 | 33 | 31 | 22 |
| **DEGL** | 24 | 31 | 17 | 31 | 30 | 29 | 14 | 15.5 | 25 | 25 | 25 | 7 | 24 | 11 | 11 | 11 | 25 | 28 | 27 | 28 | 9 | 21 |
| **DE-SG** | 14 | 21 | 17 | 29 | 13 | 9 | 11 | 15.5 | 28 | **1** | 31 | 31 | 13 | 14 | 19 | 10 | 9 | 9 | 14 | 8 | 14 | 8 |
| **EPSDE** | 4 | 27 | 17 | 24 | 25 | 21 | 30 | 15.5 | 19 | 16 | 23 | 15 | 11 | 15 | 13 | 16 | 12 | 16 | 15 | 13 | 28 | 5 |
| **FDSADE** | 23 | 22 | 17 | 27 | 24 | 23 | 8 | 15.5 | 18 | 10 | 13 | 11 | 23 | 23 | 17 | 25 | 19 | 18 | 23 | 24 | 24 | 12 |
| **GA-MPC** | 17 | **3** | 17 | **3** | 20 | 18 | 24 | 15.5 | 9 | 4 | 17 | **1** | **2** | 16 | 5 | 13 | 26 | 25 | 17 | 26 | **2** | 6 |
| **GLPSO** | 28 | 15 | 17 | 15 | 19 | 20 | 16 | 15.5 | 23 | 23 | 14 | 24 | 31 | 30 | 30 | 29 | 23 | 19 | 20 | 19 | 19 | 27 |
| **HCLPSO** | 8 | 7 | 17 | 7 | 9 | 12 | 17 | 15.5 | 5 | 17 | 15 | 12 | 14 | 26 | 26 | 23 | 15 | 14 | 22 | 17 | 4 | 23 |
| **jDEIscop** | 4 | 17 | 17 | 33 | 5 | 6 | 19 | 15.5 | 7 | 6 | 26 | 5 | 8 | 10 | **3** | 9 | 22 | 23 | 24 | 23 | 16 | **1** |
| **LBBO** | 4 | 4 | 17 | 5 | 6 | 17 | 6 | 15.5 | 27 | 30 | 5 | 29 | 16 | 18 | 21 | 33 | 33 | 30 | 29 | 30 | 7 | 16 |
| **MDE-pBX** | 21 | 20 | 17 | 22 | 28 | 26 | 9 | 15.5 | 17 | 9 | 11 | 16 | 20 | 6 | 12 | 8 | 7 | 5 | 7 | 9 | 21 | 20 |
| **MPEDE** | 22 | 5 | 17 | 8 | 18 | 16 | 4 | 15.5 | 6 | 21 | 4 | **3** | 18 | 8 | 7 | 4 | 4 | 4 | **2** | 4 | 10 | 10 |
| **NMA** | 32 | 8 | 17 | 6 | 17 | 28 | 7 | 15.5 | 31 | 27 | 6 | 33 | 30 | 21 | 28 | 5 | 16 | 13 | 9 | 14 | 30 | 31 |
| **PMS** | 26 | 25 | 17 | 4 | 21 | 14 | 29 | 33 | 4 | 32 | 10 | 22 | 32 | 28 | 33 | 32 | 30 | 32 | 32 | 32 | 32 | 32 |
| **PSO-iw** | 27 | 24 | 17 | 10 | 26 | 30 | 15 | 32 | 30 | 26 | 30 | 30 | 27 | 27 | 31 | 30 | 21 | 22 | 21 | 21 | 26 | 30 |
| **RA** | 33 | 29 | 17 | **1** | 16 | 19 | 27 | 31 | **3** | 33 | 27 | 23 | 22 | 20 | 32 | 31 | 24 | 26 | 28 | 25 | 33 | 33 |
| **Rcr-JADE** | 12 | 12 | 17 | 14 | 4 | **3** | 22 | 15.5 | 15 | 8 | 19 | 4 | 7 | 7 | 4 | 6 | 5 | 6 | 6 | 5 | 23 | 15 |
| **SADE** | 4 | 10 | 17 | 21 | 7 | 10 | 12 | 15.5 | 13 | 7 | 16 | 18 | 10 | 17 | 15 | 18 | 13 | 17 | 18 | 16 | 17 | 11 |
| **SapsDE** | 19 | 14 | 17 | 25 | **2** | **2** | 20 | 15.5 | 12 | 19 | 18 | 8 | 17 | 12 | 18 | 19 | 14 | 11 | 11 | 10 | 20 | 18 |
| **SFMDE** | 31 | 28 | 17 | 18 | 29 | 27 | 21 | 15.5 | 29 | 22 | 28 | 28 | 5 | 25 | 25 | 20 | 32 | 31 | 31 | 31 | 25 | 25 |
| **SPS-L-SHADEEIG** | 13 | **1** | 17 | 26 | 12 | 8 | **1** | 15.5 | 10 | **2** | **1** | 21 | 4 | **1** | **1** | **1** | **2** | **1** | 4 | **1** | **1** | **2** |
| **SP-UCI** | 29 | 32 | 17 | 28 | 33 | 33 | 33 | 15.5 | 22 | 14 | 9 | 26 | 29 | **3** | 9 | **2** | **1** | **2** | **1** | **2** | 6 | 26 |
| **SspDE** | 10 | 16 | 17 | 17 | **3** | 4 | 26 | 15.5 | 14 | 13 | 22 | 19 | **3** | 13 | 14 | 14 | 11 | 12 | 16 | 11 | 15 | 9 |

**Table 9**
22-problem averaged rankings of algorithms for various maximum numbers of function calls. The lower averaged ranking, the better algorithm. The best algorithms for each computational budget are bolded.

| Allowed number of function calls | 5000 | 20,000 | 50,000 | 100,000 | 150,000 | 500,000 |
|---|---|---|---|---|---|---|
| AdapSS-JADE | 18.4 | 16.5 | 11.5 | 11.7 | 12.1 | 13.7 |
| ALC-PSO | 10.3 | 16.2 | 20.2 | 23.2 | 23.7 | 24.7 |
| AMALGAM | 11.9 | 15.7 | 15.4 | 15.8 | 15.7 | 15.3 |
| AM-DEGL | 19.8 | 17.9 | 16.0 | 12.8 | 12.1 | 8.9 |
| ANS | 17.0 | 22.2 | 23.2 | 24.7 | 24.3 | 26.0 |
| ATPS-DE | 9.5 | 8.5 | 10.0 | 10.5 | 11.5 | 13.5 |
| CDE | 22.4 | 20.0 | 18.0 | 16.5 | 15.4 | 13.9 |
| CLPSO | 20.3 | 20.7 | 20.5 | 19.3 | 18.9 | 17.1 |
| CLPSO-DEGL | 9.4 | 15.5 | 17.0 | 19.0 | 19.2 | 20.6 |
| CoBiDE | 21.5 | 19.4 | 14.5 | 9.3 | **7.5** | 8.1 |
| DE | 28.5 | 28.4 | 28.3 | 27.4 | 27.1 | 26.6 |
| DEGL | 21.4 | 19.8 | 20.7 | 21.9 | 21.4 | 21.3 |
| DE-SG | 24.3 | 22.8 | 23.1 | 23.2 | 22.2 | 15.4 |
| EPSDE | 13.8 | 12.8 | 13.6 | 13.9 | 15.0 | 17.3 |
| FDSADE | 14.2 | 14.0 | 14.1 | 14.7 | 16.7 | 19.1 |
| GA-MPC | 20.5 | 16.7 | 15.2 | 12.6 | 12.4 | 13.0 |
| GLPSO | 11.2 | 12.9 | 14.8 | 17.4 | 18.5 | 21.7 |
| HCLPSO | 10.0 | 12.1 | 14.0 | 14.8 | 14.7 | 14.8 |
| jDElscop | 18.1 | 17.6 | 17.4 | 17.3 | 17.1 | 13.6 |
| LBBO | 12.3 | 15.3 | 17.8 | 18.3 | 18.7 | 18.1 |
| MDE-pBX | 11.8 | **8.3** | 9.5 | 11.8 | 12.3 | 14.8 |
| MPEDE | 19.7 | 17.4 | 13.7 | 10.8 | 9.3 | 9.6 |
| NMA | 20.1 | 22.5 | 22.9 | 23.5 | 22.8 | 19.3 |
| PMS | 14.0 | 19.1 | 20.6 | 22.7 | 24.1 | 25.1 |
| PSO-iw | **7.9** | 15.4 | 19.7 | 21.0 | 23.4 | 25.1 |
| RA | 18.8 | 22.5 | 24.0 | 24.4 | 24.2 | 24.2 |
| Rcr-JADE | 18.5 | 11.5 | **7.7** | **8.3** | 8.4 | 10.4 |
| SADE | 13.2 | 10.9 | 11.6 | 12.0 | 12.5 | 13.8 |
| SapsDE | 13.7 | 9.7 | 10.4 | 11.6 | 13.0 | 14.6 |
| SFMDE | 16.9 | 21.4 | 23.5 | 24.3 | 24.2 | 24.7 |
| SPS-L-SHADEEIG | 20.4 | 16.7 | 14.9 | 11.5 | 10.4 | **6.6** |
| SP-UCI | 29.0 | 26.5 | 24.4 | 21.7 | 20.0 | 16.9 |
| SspDE | 22.2 | 14.2 | 12.7 | 13.1 | 12.3 | 13.4 |

**Table 10**
Final place in each competition based on averaged ranks from Table 9.

| Allowed number of function calls | 5000 | 20,000 | 50,000 | 100,000 | 150,000 | 500,000 |
|---|---|---|---|---|---|---|
| AdapSS-JADE | 18 | 16 | 5 | 7 | 6 | 10 |
| ALC-PSO | 5 | 15 | 23 | 27 | 28 | 28 |
| AMALGAM | 8 | 14 | 16 | 16 | 16 | 16 |
| AM-DEGL | 22 | 21 | 17 | 11 | 7 | 3 |
| ANS | 16 | 28 | 29 | 32 | 32 | 32 |
| ATPS-DE | 3 | 2 | 3 | 3 | 5 | 8 |
| CDE | 30 | 25 | 21 | 17 | 15 | 12 |
| CLPSO | 24 | 26 | 24 | 22 | 21 | 19 |
| CLPSO-DEGL | 2 | 13 | 18 | 21 | 22 | 24 |
| CoBiDE | 28 | 23 | 12 | 2 | 1 | 2 |
| DE | 32 | 33 | 33 | 33 | 33 | 33 |
| DEGL | 27 | 24 | 26 | 25 | 24 | 25 |
| DE-SG | 31 | 31 | 28 | 28 | 25 | 17 |
| EPSDE | 12 | 7 | 8 | 13 | 14 | 20 |
| FDSADE | 14 | 9 | 11 | 14 | 17 | 22 |
| GA-MPC | 26 | 18 | 15 | 10 | 10 | 6 |
| GLPSO | 6 | 8 | 13 | 19 | 19 | 26 |
| HCLPSO | 4 | 6 | 10 | 15 | 13 | 14 |
| jDElscop | 17 | 20 | 19 | 18 | 18 | 9 |
| LBBO | 9 | 11 | 20 | 20 | 20 | 21 |
| MDE-pBX | 7 | 1 | 2 | 8 | 8 | 15 |
| MPEDE | 21 | 19 | 9 | 4 | 3 | 4 |
| NMA | 23 | 29 | 27 | 29 | 26 | 23 |
| PMS | 13 | 22 | 25 | 26 | 29 | 30 |
| PSO-iw | 1 | 12 | 22 | 23 | 27 | 31 |
| RA | 20 | 30 | 31 | 31 | 31 | 27 |
| Rcr-JADE | 19 | 5 | 1 | 1 | 2 | 5 |
| SADE | 10 | 4 | 6 | 9 | 11 | 11 |
| SapsDE | 11 | 3 | 4 | 6 | 12 | 13 |
| SFMDE | 15 | 27 | 30 | 30 | 30 | 29 |
| SPS-L-SHADEEIG | 25 | 17 | 14 | 5 | 4 | 1 |
| SP-UCI | 33 | 32 | 32 | 24 | 23 | 18 |
| SspDE | 29 | 10 | 7 | 12 | 9 | 7 |

**Table 11**
General applicability of algorithms. In columns the number of wins by each algorithm for particular computational budget, the number of times each algorithm is within best 3, or best 5 optimizers is shown. Additionally, the correlation coefficients between ranks achieved for 22 problems in competitions with 5000 and 150,000 function calls, and in competitions with 150,000 and 500,000 function calls is given.

| Algorithms | Number of wins | | | | | | Number of times within best 3 | | | | | | Number of times within best 5 | | | | | | Correlation between ranks achieved in competitions with 5000 and 150,000 function calls | Correlation between ranks achieved in competitions with 150,000 and 500,000 function calls |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Allowed number of function calls | 5000 | 20,000 | 50,000 | 100,000 | 150,000 | 500,000 | 5000 | 20,000 | 50,000 | 100,000 | 150,000 | 500,000 | 5000 | 20,000 | 50,000 | 100,000 | 150,000 | 500,000 | | |
| AdapSS-JADE | **0** | **0** | **1** | **0** | **0** | **1** | 0 | 0 | 2 | 2 | 2 | 1 | **0** | **0** | **4** | **5** | **5** | **2** | 0.29 | **0.88** |
| ALC-PSO | **1** | **0** | **0** | **0** | **0** | **0** | 6 | 0 | 1 | 1 | 1 | 1 | **9** | **1** | **1** | **1** | **1** | **1** | −0.20 | **0.91** |
| AMALGAM | **1** | **1** | **1** | **1** | **1** | **0** | 4 | 3 | 3 | 2 | 3 | 3 | **5** | **4** | **3** | **4** | **5** | **3** | 0.27 | **0.90** |
| AM-DEGL | **0** | **0** | **0** | **0** | **0** | **1** | 1 | 1 | 0 | 2 | 1 | 8 | **2** | **1** | **1** | **3** | **2** | **11** | 0.22 | **0.62** |
| ANS | **0** | **0** | **0** | **0** | **0** | **0** | 0 | 0 | 1 | 0 | 1 | 0 | **2** | **0** | **1** | **0** | **1** | **0** | 0.36 | **0.90** |
| ATPS-DE | **0** | **0** | **0** | **0** | **0** | **0** | 2 | 2 | 3 | 2 | 0 | 0 | **8** | **9** | **4** | **4** | **3** | **0** | 0.26 | **0.86** |
| CDE | **0** | **0** | **0** | **0** | **0** | **1** | 0 | 0 | 0 | 2 | 3 | 3 | **0** | **1** | **2** | **2** | **3** | **4** | −0.32 | **0.79** |
| CLPSO | **0** | **0** | **0** | **0** | **0** | **0** | 1 | 0 | 0 | 0 | 0 | 1 | **1** | **0** | **0** | **0** | **0** | **1** | 0.44 | **0.81** |
| CLPSO-DEGL | **0** | **0** | **0** | **0** | **0** | **0** | 7 | 0 | 1 | 1 | 0 | 0 | **7** | **0** | **1** | **1** | **0** | **0** | 0.37 | **0.77** |
| CoBiDE | **0** | **0** | **0** | **1** | **2** | **1** | 0 | 0 | 1 | 8 | 10 | 5 | **0** | **1** | **2** | **9** | **14** | **10** | −0.05 | **0.79** |
| DE | **0** | **0** | **0** | **0** | **0** | **0** | 0 | 0 | 0 | 0 | 1 | 0 | **0** | **0** | **0** | **0** | **1** | **1** | 0.72 | **0.97** |
| DEGL | **0** | **0** | **0** | **0** | **0** | **0** | 1 | 1 | 1 | 0 | 0 | 0 | 1 | **1** | **2** | **1** | **0** | **0** | 0.34 | **0.73** |
| DE-SG | **0** | **0** | **0** | **0** | **0** | **1** | 0 | 0 | 0 | 0 | 0 | 1 | **1** | **0** | **0** | **0** | **0** | **1** | 0.41 | **0.54** |
| EPSDE | **0** | **1** | **0** | **0** | **0** | **0** | 1 | 5 | 2 | 0 | 0 | 0 | **3** | **6** | **4** | **4** | **1** | **2** | 0.54 | **0.84** |
| FDSADE | **0** | **0** | **0** | **0** | **0** | **0** | 0 | 1 | 1 | 1 | 0 | 0 | **2** | **2** | **2** | **2** | **1** | **0** | −0.04 | **0.85** |
| GA-MPC | **1** | **3** | **4** | **4** | **4** | **1** | 1 | 4 | 5 | 5 | 4 | 5 | **2** | **5** | **5** | **6** | **6** | **7** | −0.06 | **0.82** |
| GLPSO | **0** | **0** | **0** | **1** | **0** | **0** | 0 | 4 | 3 | 1 | 1 | 0 | **6** | **6** | **3** | **1** | **1** | **0** | 0.52 | **0.90** |
| HCLPSO | **0** | **1** | **1** | **0** | **0** | **0** | 4 | 3 | 1 | 2 | 2 | 0 | **9** | **4** | **5** | **3** | **3** | **2** | 0.37 | **0.93** |
| jDElscop | **0** | **0** | **1** | **0** | **0** | **1** | 2 | 0 | 1 | 1 | 2 | 2 | **3** | **1** | **2** | **2** | **2** | **5** | 0.75 | **0.84** |
| LBBO | **2** | **2** | **2** | **3** | **1** | **0** | 5 | 4 | 3 | 3 | 3 | 0 | **8** | **5** | **5** | **4** | **4** | **4** | 0.68 | **0.92** |
| MDE-pBX | **2** | **4** | **0** | **0** | **0** | **0** | 2 | 7 | 3 | 1 | 1 | 0 | **3** | **10** | **10** | **5** | **5** | **1** | 0.25 | **0.77** |
| MPEDE | **0** | **0** | **0** | **1** | **4** | **0** | 0 | 2 | 1 | 3 | 5 | 2 | **2** | **3** | **3** | **5** | **7** | **9** | −0.37 | **0.76** |
| NMA | **0** | **0** | **0** | **0** | **0** | **0** | 2 | 1 | 1 | 2 | 0 | 0 | **4** | **2** | **2** | **3** | **3** | **1** | 0.41 | **0.70** |
| PMS | **3** | **3** | **1** | **1** | **0** | **0** | 4 | 6 | 5 | 2 | 1 | 0 | **6** | **6** | **5** | **3** | **2** | **2** | 0.78 | **0.98** |
| PSO-iw | **5** | **0** | **0** | **0** | **0** | **0** | 7 | 1 | 0 | 0 | 0 | 0 | **10** | **3** | **1** | **1** | **0** | **0** | 0.49 | **0.90** |
| RA | **0** | **0** | **0** | **0** | **2** | **1** | 3 | 0 | 0 | 1 | 2 | 2 | **4** | **2** | **0** | **2** | **2** | **2** | 0.42 | **0.97** |
| Rcr-JADE | **0** | **2** | **5** | **3** | **0** | **0** | 1 | 2 | 11 | 7 | 7 | 1 | **1** | **6** | **12** | **9** | **10** | **6** | 0.40 | **0.87** |
| SADE | **4** | **0** | **0** | **0** | **0** | **0** | 4 | 3 | 0 | 1 | 0 | 0 | **5** | **4** | **2** | **3** | **2** | **1** | −0.06 | **0.74** |
| SapsDE | **0** | **0** | **1** | **1** | **2** | **0** | 1 | 3 | 3 | 2 | 2 | 2 | **2** | **7** | **7** | **3** | **2** | **2** | 0.04 | **0.90** |
| SFMDE | **0** | **0** | **0** | **0** | **0** | **0** | 0 | 0 | 0 | 0 | 0 | 0 | **0** | **0** | **0** | **0** | **0** | **1** | −0.03 | **0.94** |
| SPS-L-SHADE-EIG | **1** | **0** | **1** | **1** | **3** | **9** | 1 | 1 | 3 | 4 | 8 | 12 | **2** | **3** | **4** | **8** | **8** | **14** | 0.42 | **0.70** |
| SP-UCI | **0** | **1** | **2** | **2** | **0** | **2** | 0 | 2 | 2 | 2 | 1 | 6 | **0** | **2** | **3** | **3** | **3** | **6** | −0.02 | **0.88** |
| SspDE | **0** | **2** | **0** | **0** | **0** | **0** | 0 | 4 | 2 | 2 | 1 | 2 | **1** | **5** | **4** | **3** | **3** | **3** | 0.01 | **0.81** |

From Tables 9 and 10 we may also note that apart from the basic DE, which (at least with parameter settings used) is constantly the worst or the second worst method, the 22-problem averaged ranks of the vast majority of metaheuristics depend noticeably on the computational budget. Relatively low variability of averaged ranks is noted for four algorithms only, namely AMALGAM (whose ranks range from 8th to 16th), CLPSO (19th to 26th), DEGL (24th to 27th) and NMA with re-initialization (23rd to 29th). Many algorithms constantly improve or deteriorate with increasing MNFC. For example SP-UCI with 10 complexes is the worst with the lowest computational budget – although, as discussed in Section 2, it gains some extra calls – but improves steadily when the MNFC increases and is ranked 18th when MNFC = 500,000. Similar trends are observed for various DE algorithms, including DE-SG, SPS-L-SHADE-EIG, CDE and AM-DEGL, as well as GA-MPC. The opposite relation between the averaged ranks and MNFC is noted for most of the considered PSO-based variants (PSO-iw is 1st when MNFC = 5000, but places 31st when MNFC = 500,000, a similar deterioration in averaged ranking is also noted for ALC-PSO – 5th to 28th, GLPSO – 6th to 26th and CLPSO-DEGL – 2nd to 24th) and a number of novel kinds of metaheuristics (LBBO – 9th to 21st, ANS – 16th to 32nd, PMS – 13th to 30th). There are also algorithms that perform best when MNFC is set to intermediate values, and show relatively worse performance when MNFC is either very low or high. Such methods mainly include the rest of control-parameters adaptive DE variants, namely AdapSS-JADE and Rcr-JADE (both receive best ranks when MNFC = 50,000–150,000), MDE_pBX, SADE and SapsDE (all three are favorably ranked when MNFC = 20,000–50,000). We may suppose that the adaptation of control parameters of such methods plays a vital role unless the population settles down to some local optima; after that the adaptation loses its advantage and algorithms do not improve faster than their competitors. Adaptive memetic DE variants that occasionally use local searchers (like AM-DEGL), those that use clustering (like CDE), distribute the population into sub-populations (MPEDE, DE-SG), rotate coordinate systems (CoBiDE, SPS-L-SHADE-EIG), or follow linear population size reduction (SPS-L-SHADE-EIG) seem to avoid such a situation and continue improving the relative performance during the later stages of the search. We may also note that no pattern for variation of ranks of some DE (SspDE) or PSO (HCLPSO) variants is observed when MNFC is higher than 5000. Interestingly, two of the oldest methods, NMA and RA (that we use with re-initialization options, see Table 2), perform slightly better with the lowest and the highest tested MNFC than with moderate computational budgets. As the most modern metaheuristics were tuned and tested on problems with moderate values of MNFC, the competition with historical methods seems to be more "fair" when MNFC is very large or small. Nonetheless, it is worth noting that both NMA and RA, proposed during the 1960s, if coupled with a simple restart method, are still competitive to some modern metaheuristics.

Although it is hard to firmly link the relation between the MNFC and the relative performance with specific features of particular algorithms, as they differ greatly, we may note some fairy general findings.

First, let us focus on population sizes and convergence speed. In the literature (see references from Table 2) PSO algorithms are often run with lower population sizes (most frequently with 20–40 particles) than DE or GA ones (which often use at least 50 individuals), although there is no consensus regarding DE population size, which may vary in different DE variants by even more than an order of magnitude [13,45]. One should also note that the modifications of PSO [7,34,35] have mainly been proposed to prevent premature convergence, not to make convergence quicker. On the contrary, DE instead suffers from slow convergence [13]. Both highlighted factors may help explain the performance of particular types of algorithms. The results obtained in this study, based on tests performed on CEC2011 real-world problems, suggest that PSO-based metaheuristics rank relatively better when MNFC is low, since they use smaller population sizes and rather suffer from premature convergence. By contrast, DE variants are more explorative. With larger population sizes and slow convergence, DE approaches (and GA-MPC) require a lot of time to show their advantages, thus they perform relatively better when MNFC is high. The success of SPS-L-SHADE-EIG when MNFC is high, and its poor performance when MNFC is low, may be partly attributed to the fact that it linearly decreases the population size during the run, from exceptionally large (19 times the problem dimensionality) at the beginning of the search, to very low (4 individuals) at the end. As a result, when MNFC is low, the initial explorative steps of SPS-L-SHADE-EIG are quickly abandoned, wasting the function calls used for them. On the other hand, when MNFC is very high, the algorithm explores various areas of the search space detecting local optima, and gradually switches from explorative to exploitative behavior by slowly eliminating the least useful individuals in the population.

However, the impact of the population size on our results is not necessarily as simple as outlined above, which may be illustrated by two important examples. CoBiDE is the DE variant that uses quite a small population size (60 individuals), but counterintuitively it performs very poorly when MNFC is low. However, it becomes the second best among 30 metaheuristics tested when MNFC is set to the largest considered values. ATPS-DE is able to flexibly modify its population size (note that unlike SPS-L-SHADE-EIG, it allows population to both decrease and increase during the run), but such adaptation requires time. Due to that, we could expect that its relative performance should improve with increasing MNFC (at least as in the case of SPS-L-SHADE-EIG). However, in tests we find that ATPS-DE turns out better ranked when MNFC is very low than when it is high.

As one would expect, approaches that use some methods to promote diversity of the population [53], especially distributed algorithms (MPEDE, AM-DEGL, DE-SG, SP-UCI), are better ranked when MNFC is set higher. Although distributed HCLPSO, like other PSO variants, is best ranked when MNFC is low, maintaining the population diversity allows it to achieve relatively good rankings (13th – 14th) compared to other PSO variants even for the largest MNFC tested. On the other hand, the relatively better performance of AMALGAM and LBBO when MNFC is low may be attributed to the fact that such hybrids rely on local search methods – AMALGAM on CMA-ES [26], and LBBO on gradient descent [51]. This speeds up convergence, but may affect exploration. ANS and PMS are also better ranked when MNFC is low, probably because both approaches seem

to lack an effective exploration mechanism, hence the way that they are searching for a new basin of attraction seems too randomized. Interestingly, although PMS often outperforms its much older antecessor, RA, this is not the case in tests where the MNFC is set to 500,000. On the other hand, SP-UCI, which is based on NMA experience, demonstrates its advantage over the ancestor only when MNFC is sufficiently large, which may be explained by both the large population size of the SP-UCI version used and the fact that algorithmic elements that cause SP-UCI to differ from NMA are exploration-oriented.

Six of the tested algorithms use variable population size (AMALGAM, ATPS-DE, FDSADE, jDElscop, SapsDE and SPS-L-SHADE-EIG), but excluding SPS-L-SHADE-EIG this property does not seem to impact the relation between the relative performance and MNFC. For these five algorithms the population size variability may not be a decisive factor in gaining the best performance, but all of them have been ranked overall among best two thirds (67%) of methods in each of six competitions. This suggests that variable population size most likely helps to avoid pitfalls. Interestingly, the linear population size reduction method used by SPS-L-SHADE-EIG turns out very successful for higher computational budgets, but at the cost of the poor performance when the MNFC is very low.

To summarize, there is quite a clear relation between the computational budget and the relative performance of metaheuristics. The majority of the tested PSO variants and novel kinds of optimizers are ranked better on CEC2011 real-world problems when MNFC is set low. On the other hand, a large proportion of DE algorithms, as well as GA-MPC and complex-based SP-UCI, are ranked better when MNFC is large. The best ranked algorithms based on results under very low MNFC conditions perform poorly when MNFC is set to be high, and vice versa. Only one algorithm, ATPS-DE, performs relatively well with each MNFC, however it is never ranked best (its position according to a 22-problems averaged ranking varying with computational budget is between second and eighth best), and it seems better suited to low-to-moderate computational budgets (as it is ranked 8th when MNFC is set to the highest considered value).

### *3.2. Specialization versus general applicability of metaheuristics with respect to the computational budget*

Averaged ranks may help determining some general relations, but do not allow one to find which algorithms are more specialized and indeed how often they triumph for specific problems. A closer, problem-related examination is needed to find algorithms that win competitions for numerous problems, algorithms that often perform well even though they rarely win, and those that may fully fail on specific problems.

First, let us briefly examine the results on particular problems given in Figs. 1–22 and the respective problem-related rankings (Tables 3-8). From Figs. 1–22 one sees that the performances of all 30 algorithms are truly uneven on most problems (with rare exceptions, e.g. Fig. 3), and that they highly depend on the MNFC set. On some problems one or a few algorithms fully fail, performing even by orders of magnitude poorer than the majority of approaches. Such failures may occur for the specific computational budgets, or irrespectively of the MNFC. Algorithms that occasionally perform outstandingly poorly include PMS on problem P8 (and RA when MNFC is low), RA on P10 (but only when MNFC is low), DE on P11.1, NMA and RA on P11.3 (but only when MNFC = 5000), RA and SP-UCI on P11.5 (when MNFC is low), SP-UCI, NMA and DE-SG on P11.6 (but only when MNFC = 5000), LBBO and SFMDE on P11.7 (when MNFC is large), DE, PMS and SFMDE on P11.8 and P11.10 (when MNFC is large), RA on P12 and, when MNFC is low, P13. Hence, we observe that eight algorithms, namely RA, PMS, NMA, DE, SP-UCI, DE-SG, LBBO and SFMDE in some cases perform much poorer than the rest of the pack. This warns us that some algorithms may be unreliable for specific real-world problems, and this unreliability may, but does not have to, depend on the computational budget. Interestingly, the list does not include any PSO-based approaches tested, and the problems on which some algorithms fail in comparison with the majority of others may be both low- and high-dimensional, and may come from various fields of science.

On the contrary, visual inspection of Figs. 1–22 shows that one or two algorithms would rarely perform by far better than all other metaheuristics on a specific problem. The only case when one algorithm clearly dominates all others is GA-MPC on problem P11.2 (perhaps because P11.2 is the highest-dimensional problem considered).

The runs performed for each MNFC are independent and therefore the results achieved by a particular algorithm with the larger MNFC are occasionally poorer than those with the lower one – for example note the performances of AM-DEGL, ANS, CLPSO-DEGL or DE-SG on P1 when MNFC is set to 150,000 and 500,000 (see Figs. 1–22). These phenomena reflect the fact that real-world problems often have plenty of local optima (in P1 the objective function represents the sum of square errors between estimated and target data), and some algorithms frequently prematurely converge to one of them, or become stagnant [13]. In such cases the remaining function calls are simply wasted. Hence, if by chance an algorithm more often prematurely converges in runs with higher MNFC than in runs with lower MNFC, it may achieve poorer results in the former case. Obviously, as may be seen in Figs. 1–22, it often happens when one compares the results achieved after the two highest MNFC values.

From Figs. 1–22, Table 11 and detailed Tables 3–8 we may find that, with the exception of GA-MPC on P11.2 which has already been discussed, no algorithm outstandingly outperforms all others. However, for most problems and numbers of function calls there is a single winner, i.e. the algorithm that on average performs better than the others with respect to the threshold level that has been set in Section 2. We do not refer herein to the statistical significance that will be addressed in the further part of the present section. From Table 11 we may find that different algorithms achieve the largest number of wins in competitions with low and high MNFC. When MNFC is set to the lowest considered value (5000), PSO-iw wins for five out of 22 problems, including both 1- and 96-dimensional ones; followed by SADE (four wins, on problems of various dimensionality, but only those representing economic load dispatch tasks) and PMS (three wins, on higher-dimensional

problems). Most algorithms, including CLPSO-DEGL and ATPS-DE that were, according to 22-problems averaged ranks, considered 2nd and 3rd best for MNFC = 5000, do not win for any problem. This shows that all-problems averaged ranks do not necessarily point to methods that perform best for any specific problem; they may simply point to algorithms that on the vast majority of problems show sufficiently good performance. Indeed, as we find in Table 11 for MNFC set to 5000, ATPS-DE, although never the best and just twice within the best three, was for 8 out of 22 problems among the five best methods, being frequently 4th or 5th (interestingly, none of such problems was from physics, chemistry or other fields of natural sciences). CLPSO-DEGL is ranked among the best three methods on seven problems (30% of all considered ones, mostly from engineering areas), but it never wins. But the method with the best averaged rank, PSO-iw, is ranked among the best five algorithms on a similar number of problems (10), which are from various fields of science and are of differentiated dimensionalities.

Remarkably, when MNFC is set to 20,000, MDE-pBX is the most frequent winner (four wins, on 6- to 96-dimensional problems from spacecraft trajectory optimization, hydrothermal scheduling and sound waves modeling), followed by GA-MPC (three wins, but on higher-dimensional problems), but PSO-iw does not rank first on any problem. This shows how quickly the relative performance of algorithms may change when MNFC is set low.

For moderate MNFC values (50,000–100,000), Rcr-JADE, GA-MPC, LBBO and SP-UCI are most frequent winners. We find that Rcr-JADE is most often noted among the best five methods, followed by MDE-pBX. Rcr-JADE does not win for any problem when MNFC is increased to 150,000, although it is still frequently ranked among the best five methods. Overall, when MNFC is set to 50,000–150,000, Rcr-JADE performs best for problems of various dimensionalities, but only from the fields of economics, hydraulics and engineering – it cannot be recommended for those from natural sciences or spacecraft trajectory optimization. On the other hand, GA-MPC performs best for moderate- or high-dimensional problems and the performances of LBBO and MDE-pBX are not related either to problem dimensionality or to any scientific field.

In competitions with the largest MNFC value (500,000) the situation changes. SPS-L-SHADE-EIG emerges as the best for nine out of 22 problems. These nine problems are all at least 10-dimensional, but originate from different fields of science. The only other method that gains more than a single win when MNFC is set to 500,000 is SP-UCI, although it remains in the middle of the flock according to averaged ranks (see Tables 9 and 10). Indeed, SP-UCI seems to be a very specialized method – it is among two best algorithms on six problems, but (see Table 8) for three problems, all of which are 20- or 30-dimensional and arise from areas of physics or communication, it is noted as the poorest method, and on many others it is ranked within the worst 10. SPS-L-SHADE-EIG is among 5 best approaches on the majority of problems (14 in total) which come from various fields of science; but interestingly, it performs rather poorly on the 1-dimensional problems 3 and 4. AM-DEGL is ranked among the five best methods on 11 problems (with various dimensionalities; most such problems are related to economy), CoBiDE on 10 (generally those that are not high-dimensional) and MPEDE on 9 problems (which don't share any problem dimensionality commonalities, but rather all originate from fields that are not from natural sciences). One may further note that all such algorithms are also ranked high according to the averaged ranks (see Tables 9 and 10). Various other algorithms, including GA-MPC, Rcr-JADE and SP-UCI are also noted among the best five methods on around six to seven problems each.

Considering together the results achieved for various computational budgets, we note that many algorithms receive no wins in any competition, but extremely simple, non-population based historical RA (equipped with the option to restart) emerges as the best for two problems (both 1-dimensional P4 and large-dimensional P9) when 150,000 function calls are allowed, and moreover for one (P4) when MNFC is equal to 500,000 (see Table 11). This raises a question as to why all of the 30 + modern heuristics fail on such real-world problems, but it also shows that for some applications the historical approaches may still be competitive with the current metaheuristics, and sometimes may even become the best choice. Such a result may agree with the findings from [15,62] that, although obtained for various kinds of problems and competing frameworks, also revealed that the metaheuristics used were not as good as one may wish to see for performances of modern methods on specific tasks. On the other hand, each algorithm has been noted at least once among the best five methods. This may suggest that if one keeps searching long enough, any metaheuristic may find its practical applications. The problem of whether this justifies its existence [52] is beyond the scope of this paper.

Table 11 confirms that some algorithms perform much better when MNFC is low, others when MNFC is high. CoBiDE is among the five best methods for 40% to 63% of problems when MNFC is set to at least 100,000, but is never among the best five when it is set to 5000, and just once when MNFC is equal to 20,000. Moreover, SPS-L-SHADE-EIG is among the five best performing methods for 63% of problems when MNFC is set to 500,000, but only for 9% when the MNFC is equal to 5000. AM-DEGL is among the five best metaheuristics for 50% of problems when MNFC is set to 500,000, but only for between 4% and 14% of problems when NMFC is somewhere between 5000 and 150,000. On the other hand, PSO-iw and ATPS-DE are among the five best methods for 36% to 45% of problems when MNFC is set to 5000, but are never within the best five when MNFC is equal to 500,000. A similar trend is observed for ALC-PSO and HCLPSO, even though these two methods are among best five methods on at least one problem for each considered MNFC.

ATPS-DE is the only algorithm that is, according to 22-problems averaged-ranks, classified among the best 10 methods for each considered MNFC. Moreover, when the MNFC is high (set to 500,000) the rankings of ATPS-DE are especially consistent for each considered problem – it is at best eighth, but for no problem worse than 20th (see Table 8), which surprisingly suffices for it to slip into 8th position according to the 22-problems averaged ranking (Table 10).

It is interesting to find a correlation between the 22 ranks that a particular algorithm achieves on all problems when MNFC is set to 5000 and those for MNFC equal to 150,000 (value suggested in CEC2011). It turns out that such correlation

heavily depends on the algorithm. For the majority of algorithms the correlation is positive but weak, however for some (MPEDE, CDE and to a lesser degree ALC-PSO) it turns out to be clearly negative. This means that these three algorithms are, when MNFC is high, better ranked on problems on which they perform relatively worse when MNFC is low. In case of MPEDE and CDE such behavior may be attributed to the fact that quite slow convergence (hence relatively poor performance after a low number of function calls) and large exploration allow such algorithms to find areas of attraction for better solutions that can be explored when MNFC is large. On the other hand, if these algorithms converge quicker on some problems, they lose diversity and are unable to find areas of attraction of better solutions. Such a negative correlation among relative performances may need more attention in the future.

The correlations among relative performances of algorithms on problems when MNFC is set to 150,000 and 500,000 are always high. Hence, once MNFC exceeds a certain threshold, the relative performance of algorithms remains at a fairly steady state. There are, however, a very small number of exceptions to this rule – these correlations are clearly lower for DE-SG and AM-DEGL algorithms than for the others. These two algorithms, together with SPS-L-SHADE-EIG (for which the respective correlation is the third lowest) achieve large relative improvements on a number of problems when the MNFC is increased from 150,000 to 500,000, which is also reflected in the averaged rankings (see Table 10).

Finally, we may ask about the statistical significance of the pairwise differences between the results achieved by various algorithms. These (for the methodology used see Section 2) are detailed in Tables 12–17, separately for each computational budget. As we see, most pairwise comparisons are not statistically significant, but the differences between the best algorithms and a large number of their competitors are significant at $\alpha = 0.05$. The same may be observed for pairwise comparisons between the worst algorithms and most of their competitors. By counting the number of times each algorithm is statistically better or worse according to the pairwise comparisons, (see Table 18), we find a good consistency with the results discussed in Section 3.1. The basic DE variant is significantly poorer than at least 50% of competitors, irrespective of the MNFC, and is never statistically better than any algorithm. PSO-iw is significantly better than 13 algorithms when MNFC is equal to 5000, but significantly worse than 11 algorithms when MNFC is set to 500,000. DE, ANS, PMS and PSO-iw are among the algorithms that are statistically significantly worse than many of the competing methods when MNFC is set to 500,000, with 16, 13, 11 and 11 losses respectively in pairwise comparisons. On the other hand, when MNFC is equal to 500,000 SPS-L-SHADE-EIG is statistically better than 13 algorithms, CoBiDE and AM-DEGL – 10, MPEDE – 9, Rcr-JADE – 7, GA-MPC and SspDE – 6. A number of algorithms (AdapSS-JADE, AMALGAM, ATPS-DE, EPSDE, HCLPSO, jDElscop, MDE-pBX, Rcr-JADE, SADE, SapsDE) were never significantly worse than any competitor. If we unofficially summarize the numbers of algorithms from all experiments from which a particular method is statistically significantly better, we find the largest numbers of such cumulative wins for ATPS-DE and Rcr-JADE (45 and 43, respectively), which could hence be called the most frequent winners overall. However, in general the conclusions from Table 18 that are based on statistical significance analysis confirm our previous findings.

### 3.3. Example of true computational runtime analysis on a few selected problems

In the above discussion the computational runtime of the implemented code of specific algorithms has not been considered because it depends on too many factors independent of the algorithm itself. However, for some practitioners the question may be of interest whether, at least in a specific environment, the algorithms whose code requires substantially more computational runtime lead to better results. As suggested at the end of Section 2, we finish our discussion by testing runtime of codes of all applied algorithms on three selected problems, P1, P8 and P11.6, with MNFC set to 20,000, when each code is run one by one on the same machine, 30 times for each problem.

The obtained results are summarized in Table 19. The main finding is that for a specific problem most codes consume a similar runtime. No one code requires noticeably less runtime than the majority of others.

However, some codes require even five times more computational runtime to proceed than the fastest ones. In Table 19 codes that require more than 50% runtime than the quickest one are bolded. Although the specific results do differ for the three considered problems, generally the same algorithms require more runtime in each among the tested cases; examples include AMALGAM, LBBO, ANS (for all three problems the computational time of the codes of those three algorithms is at least 50% higher than the computational time of the fastest code), and – to a lesser degree – SP-UCI, NMA, CLPSO and CLPSO-DEGL. The code excerpts for all three algorithms that require the longest runtime (AMALGAM, LBBO, ANS) have been obtained from their authors or respective web pages. The longer runtimes of AMALGAM and LBBO may be explained, as these are more complicated algorithms that include various subroutines (restart options, solutions sorting, multi-algorithm sharing, local search procedures, etc.). On the other hand, ANS is among the simplest algorithms tested. Among other "slow" codes, SP-UCI and CLPSO-DEGL hybrid are relatively complicated, but NMA and CLPSO are rather simple. Hence, from our results practitioners may find that the runtime of a specific code may in fact not be properly inferred from the simplicity of the algorithm that is seen in the paper text.

Probably more important is that the best algorithms are not those which have code that require more runtime (we refer here to Tables 4 and 19). When MNFC is set to 20,000, on problems P1 and P11.6 the best performing algorithms are those which have code that require low computational runtime (MDE-pXB, GA-MPC and SapsDE on P1, SspDE, EPSDE and SADE on P11.6). On P8 most algorithms perform equally well, but among those that perform poorer one finds that AMALGAM, CLPSO-DEGL, NMA and SP-UCI require more computational runtime than the majority of codes. Also, the codes that need

**Table 12**
The statistical significance of pair-wise comparisons among all algorithms with number of function calls set to 5000 by means of Friedman test with post-hoc Shaffer's procedure at 5% significance level. 1 – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

| 5000 function calls | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AdapSS-JADE | ALC-PSO | AMALGAM | AM-DEGL | ANS | ATPS-DE | CDE | CLPSO | CLPSO-DEGL | CoBiDE | DE | DEGL | DE-SG | EPSDE | FDSADE | GA-MPC | GLPSO | HCLPSO | jDElscop | LBBO | MDE-pBX | MPEDE | NMA | PMS | PSO-iw | RA | Rcr-JADE | SADE | SapsDE | SFMDE | SPS-L-SHADE-EIG | SP-UCI | SspDE |
| AdapSS-JADE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ALC-PSO | | | | | | | 1 | | | | 1 | | 1 | | | | | | | | | | | | | | | | | | | | 1 | 1 | |
| AMALGAM | | | | | | | | | | | 1 | | 1 | | | | | | | | | | | | | | | | | | | | 1 | | |
| AM-DEGL | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| ANS | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| ATPS-DE | | | | | | | 1 | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | 1 | 1 | |
| CDE | | 1 | | | | 1 | | | 1 | | | | | | | | | 1 | | | | | | | 1 | | | | | | | | | | |
| CLPSO | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| CLPSO-DEGL | | | | | | | 1 | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | 1 | 1 | |
| CoBiDE | | | | | | 1 | | | 1 | | | | | | | | | 1 | | | | | | | 1 | | | | | | | | | | |
| DE | | 1 | 1 | | 1 | 1 | | | 1 | | | | | 1 | 1 | | 1 | 1 | | 1 | 1 | | | 1 | 1 | | | 1 | 1 | 1 | | | | | |
| DEGL | | | | | | 1 | | | 1 | | | | | | | | | 1 | | | | | | | 1 | | | | | | | | | | |
| DE-SG | | 1 | 1 | | | 1 | | | 1 | | | | | | | | 1 | 1 | | 1 | 1 | | | | 1 | | | | | | | | | | |
| EPSDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| FDSADE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| GA-MPC | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| GLPSO | | | | | | | | | | | 1 | | 1 | | | | | | | | | | | | | | | | | | | | 1 | | | |
| HCLPSO | | | | | | | 1 | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | 1 | 1 | |
| jDElscop | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LBBO | | | | | | | | | | | 1 | | 1 | | | | | | | | | | | | | | | | | | | | 1 | | | |
| MDE-pBX | | | | | | | | | | | 1 | | 1 | | | | | | | | | | | | | | | | | | | | 1 | | | |
| MPEDE | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| NMA | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| PMS | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| PSO-iw | | | | 1 | | | 1 | 1 | | 1 | 1 | 1 | 1 | | | 1 | | | | | | 1 | 1 | | | | | | | | 1 | 1 | 1 | | |
| RA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Rcr-JADE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SADE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| SapsDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| SFMDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | |
| SPS-LSHADE-EIG | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| SP-UCI | | 1 | 1 | | 1 | 1 | | | 1 | | | | | 1 | 1 | | 1 | 1 | | 1 | 1 | | | 1 | 1 | | | 1 | 1 | 1 | | | | | |
| SspDE | | 1 | | | | 1 | | | 1 | | | | | | | | | 1 | | | | | | | 1 | | | | | | | | | | |

**Table 13**
The statistical significance of pair-wise comparisons among all algorithms with number of function calls set to 20,000 by means of Friedman test with post-hoc Shaffer's procedure at 5% significance level. 1 – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

| | 20,000 function calls | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AdapSS-JADE | ALC-PSO | AMALGAM | AM-DEGL | ANS | ATPS-DE | CDE | CLPSO | CLPSO-DEGL | CoBiDE | DE | DEGL | DE-SG | EPSDE | FDSADE | GA-MPC | GLPSO | HCLPSO | jDElscop | LBBO | MDE-pBX | MPEDE | NMA | PMS | PSO-iw | RA | Rcr-JADE | SADE | SapsDE | SFMDE | SPS-L-SHADE-EIG | SP-UCI | SspDE | | | | |
| AdapSS-JADE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ALC-PSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AMALGAM | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AM-DEGL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ANS | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | | | | | | | | | |
| ATPS-DE | | | | | 1 | | 1 | 1 | | | 1 | 1 | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | |
| CDE | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | |
| CLPSO | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | |
| CLPSO-DEGL | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CoBiDE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DE | 1 | 1 | 1 | | | 1 | | | 1 | | | | | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | | | | 1 | | 1 | 1 | 1 | | 1 | | | 1 | | | |
| DEGL | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | |
| DE-SG | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | |
| EPSDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| FDSADE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| GA-MPC | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| HCLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| jDElscop | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LBBO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MDE-pBX | | | | | 1 | | 1 | 1 | | | 1 | 1 | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | |
| MPEDE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NMA | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | | | | | | | | | |
| PMS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PSO-iw | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RA | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | | | | | | | | | |
| Rcr-JADE | | | | | | | | | | | 1 | | 1 | | | | | | | | | | | | | | | | | | | 1 | | | | | |
| SADE | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | | | 1 | | | | | |
| SapsDE | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | |
| SFMDE | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | | | 1 | | | | | | | | |
| SPS-LSHADE-EIG | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SP-UCI | | | | | | 1 | | | | | | | | 1 | 1 | | 1 | 1 | | | 1 | | | | | | 1 | 1 | 1 | | | | | 1 | | | |
| SspDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | |

**Table 14**
The statistical significance of pair-wise comparisons among all algorithms with number of function calls set to 50,000 by means of Friedman test with post-hoc Shaffer's procedure at 5% significance level. 1 – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

| | 50,000 function calls | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AdapSS-JADE | ALC-PSO | AMALGAM | AM-DEGL | ANS | ATPS-DE | CDE | CLPSO | CLPSO-DEGL | CoBiDE | DE | DEGL | DE-SG | EPSDE | FDSADE | GA-MPC | GLPSO | HCLPSO | jDElscop | LBBO | MDE-pBX | MPEDE | NMA | PMS | PSO-iw | RA | Rcr-JADE | SADE | SapsDE | SFMDE | SPS-L-SHADE-EIG | SP-UCI | SspDE | | | | | | | |
| AdapSS-JADE | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | | | |
| ALC-PSO | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | |
| AMALGAM | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AM-DEGL | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ANS | 1 | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | |
| ATPS-DE | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | | | |
| CDE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | |
| CLPSO-DEGL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CoBiDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DE | 1 | | 1 | 1 | | 1 | | | | 1 | | | | 1 | 1 | 1 | 1 | 1 | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | 1 | | | | | | |
| DEGL | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | | |
| DE-SG | 1 | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | |
| EPSDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDSADE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GA-MPC | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| HCLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| jDElscop | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LBBO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MDE-pBX | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | | | |
| MPEDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| NMA | 1 | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | |
| PMS | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | |
| PSO-iw | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | |
| RA | 1 | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | |
| Rcr-JADE | | 1 | | | 1 | | | 1 | | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | 1 | | | | | | | |
| SADE | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | | | |
| SapsDE | | | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | 1 | | | | | | | |
| SFMDE | 1 | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | |
| SPS-LSHADE-EIG | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SP-UCI | 1 | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | |
| SspDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | | | | 1 | | | | | | | |

**Table 15**
The statistical significance of pair-wise comparisons among all algorithms with number of function calls set to 100,000 by means of Friedman test with post-hoc Shaffer's procedure at 5% significance level. 1 – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

| | 100,000 function calls | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AdapSS-JADE | ALC-PSO | AMALGAM | AM-DEGL | ANS | ATPS-DE | CDE | CLPSO | CLPSO-DEGL | CoBiDE | DE | DEGL | DE-SG | EPSDE | FDSADE | GA-MPC | GLPSO | HCLPSO | jDElscop | LBBO | MDE-pBX | MPEDE | NMA | PMS | PSO-iw | RA | Rcr-JADE | SADE | SapsDE | SFMDE | SPS-L-SHADE-EIG | SP-UCI | SspDE |
| AdapSS-JADE | | 1 | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | | | |
| ALC-PSO | 1 | | | | | 1 | | | | 1 | | | | | | | | | | | 1 | 1 | | | | | 1 | | 1 | | 1 | | | |
| AMALGAM | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| AM-DEGL | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | 1 | | | | |
| ANS | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | | 1 |
| ATPS-DE | | 1 | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | | 1 | | | | 1 | | | | |
| CDE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO-DEGL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CoBiDE | | 1 | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | 1 | | |
| DE | 1 | | 1 | 1 | | 1 | | | | 1 | | | | 1 | 1 | 1 | | 1 | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | | 1 |
| DEGL | | | | | | 1 | | | | 1 | | | | | | | | | | | | | | | | | 1 | | | | | | | | |
| DE-SG | 1 | | | | | 1 | | | | 1 | | | | | | | | | | | 1 | 1 | | | | | 1 | | 1 | | 1 | | | |
| EPSDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| FDSADE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| GA-MPC | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | 1 | | | | |
| GLPSO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| HCLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| jDElscop | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LBBO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MDE-pBX | | 1 | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | | | |
| MPEDE | | 1 | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | 1 | | 1 | | | | 1 | | | | |
| NMA | 1 | | | | | 1 | | | | 1 | | | | | | | | | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | | |
| PMS | | | | | | 1 | | | | 1 | | | | | | | | | | | | 1 | | | | | 1 | | | | | | | | |
| PSO-iw | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | 1 | | | | | | | | |
| RA | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | | 1 |
| Rcr-JADE | | 1 | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | 1 | | |
| SADE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | 1 | | | 1 | | | | 1 | | | | |
| SapsDE | | 1 | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | | | |
| SFMDE | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | | 1 |
| SPS-LSHADE-EIG | | 1 | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | | | 1 | | | | 1 | | | | |
| SP-UCI | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | 1 | | | | | 1 | | | |
| SspDE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | 1 | | | | 1 | | | | |

**Table 16**
The statistical significance of pair-wise comparisons among all algorithms with number of function calls set to 150,000 by means of Friedman test with post-hoc Shaffer's procedure at 5% significance level. 1 – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

| | 150,000 function calls | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AdapSS-JADE | ALC-PSO | AMALGAM | AM-DEGL | ANS | ATPS-DE | CDE | CLPSO | CLPSO-DEGL | CoBiDE | DE | DEGL | DE-SG | EPSDE | FDSADE | GA-MPC | GLPSO | HCLPSO | jDElscop | LBBO | MDE-pBX | MPEDE | NMA | PMS | PSO-iw | RA | Rcr-JADE | SADE | SapsDE | SFMDE | SPS-L-SHADE-EIG | SP-UCI | SspDE |
| AdapSS-JADE | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | 1 | | | | 1 | | | | | | |
| ALC-PSO | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | | | | 1 | | 1 | | | |
| AMALGAM | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| AM-DEGL | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | 1 | | | | 1 | | | | | | |
| ANS | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | 1 | | | |
| ATPS-DE | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | | | | |
| CDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO-DEGL | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CoBiDE | | 1 | | | 1 | | | 1 | 1 | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | 1 | | | |
| DE | 1 | | 1 | 1 | | 1 | 1 | | | 1 | | | | 1 | | 1 | | 1 | | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | | 1 | | |
| DEGL | | | | | | | | | | 1 | | | | | | | | | | | | 1 | | | | | 1 | | | | | | | | | |
| DE-SG | | | | | | | | | | 1 | | | | | | | | | | | | 1 | | | | | 1 | | | | 1 | | | | | |
| EPSDE | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDSADE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GA-MPC | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | | 1 | | | | 1 | | | | | | |
| GLPSO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| HCLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| jDElscop | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| LBBO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MDE-pBX | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | | 1 | | | | 1 | | | | | | |
| MPEDE | | 1 | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | | | | |
| NMA | | | | | | 1 | | | | 1 | | | | | | | | | | | | 1 | | | | | 1 | | | | 1 | | | | | |
| PMS | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | | | 1 | | 1 | | | |
| PSO-iw | 1 | | | 1 | | 1 | | | | 1 | | | | | | | | | | | | 1 | | | | | 1 | | | | 1 | | | | | |
| RA | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | | | 1 | | 1 | | | |
| Rcr-JADE | | 1 | | | 1 | | | | | | 1 | 1 | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | 1 | | | |
| SADE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | | 1 | | | | 1 | | | | | |
| SapsDE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| SFMDE | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | | | 1 | 1 | | | | | 1 | 1 | | | 1 | | 1 | | | |
| SPS-LSHADE-EIG | | 1 | | | 1 | | | | | | 1 | | 1 | | | | | | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | | | |
| SP-UCI | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | 1 | | | | | | | | | |
| SspDE | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | | 1 | | | | 1 | | | | | |

**Table 17**
The statistical significance of pair-wise comparisons among all algorithms with number of function calls set to 500,000 by means of Friedman test with post-hoc Shaffer's procedure at 5% significance level. 1 – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

| 500,000 function calls | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | AdapSS-JADE | ALC-PSO | AMALGAM | AM-DEGL | ANS | ATPS-DE | CDE | CLPSO | CLPSO-DEGL | CoBiDE | DE | DEGL | DE-SG | EPSDE | FDSADE | GA-MPC | GLPSO | HCLPSO | jDElscop | LBBO | MDE-pBX | MPEDE | NMA | PMS | PSO-iw | RA | Rcr-JADE | SADE | SapsDE | SFMDE | SPS-L-SHADE-EIG | SP-UCI | SspDE |
| AdapSS-JADE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | | | | | | | | | | | | |
| ALC-PSO | | | | 1 | | | | | | 1 | | | | | | 1 | | | | | | 1 | | | | | 1 | | | | | 1 | | 1 | | | |
| AMALGAM | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AM-DEGL | | 1 | | | 1 | | | | 1 | | 1 | 1 | | | | | 1 | | | | | | | 1 | 1 | 1 | | | | 1 | | | | | | | |
| ANS | 1 | | | 1 | | 1 | 1 | | | 1 | | | | | | 1 | | | 1 | | | 1 | | | | | 1 | 1 | 1 | | 1 | | 1 | | | |
| ATPS-DE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | | | | | | | | | | | | |
| CDE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CLPSO-DEGL | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | |
| CoBiDE | | 1 | | | 1 | | | | 1 | | 1 | 1 | | | | | 1 | | | | | | | 1 | 1 | 1 | | | | 1 | | | | | | | |
| DE | 1 | | 1 | 1 | | 1 | 1 | | | 1 | | | | | | 1 | | 1 | 1 | | 1 | 1 | | | | | 1 | 1 | 1 | | 1 | | 1 | | | |
| DEGL | | | | 1 | | | | | | 1 | | | | | | | | | | | | 1 | | | | | | | | | 1 | | | | | | |
| DE-SG | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| EPSDE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDSADE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | |
| GA-MPC | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | | | | | 1 | | | | | | | |
| GLPSO | | | | 1 | | | | | | 1 | | | | | | | | | | | | 1 | | | | | | | | | 1 | | | | | | |
| HCLPSO | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| jDElscop | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | | | | | | | | | | | | |
| LBBO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | |
| MDE-pBX | | | | | | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MPEDE | | 1 | | | 1 | | | | | | 1 | 1 | | | | | 1 | | | | | | | 1 | 1 | 1 | | | | 1 | | | | | | | |
| NMA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | | | |
| PMS | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | 1 | | | 1 | | | | | 1 | 1 | | | 1 | | 1 | | | |
| PSO-iw | 1 | | | 1 | | 1 | | | | 1 | | | | | | 1 | | | 1 | | | 1 | | | | | 1 | 1 | | | 1 | | 1 | | | |
| RA | | | | 1 | | | | | | 1 | | | | | | | | | | | | 1 | | | | | 1 | | | | 1 | | | | | |
| Rcr-JADE | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | 1 | | | | 1 | | | | | | |
| SADE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | | | | | | | | | | | |
| SapsDE | | | | | 1 | | | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SFMDE | | | | 1 | | | | | | 1 | | | | | | 1 | | | | | | 1 | | | | | 1 | | | | 1 | | | | | | |
| SPS-LSHADE-EIG | | 1 | | | 1 | | | | 1 | | 1 | 1 | | | 1 | | 1 | | | 1 | | | 1 | 1 | 1 | 1 | | | | 1 | | | | | | | |
| SP-UCI | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SspDE | | 1 | | | 1 | | | | | | 1 | | | | | | | | | | | | | 1 | 1 | | | | | 1 | | | | | | | |

**Table 18**
Number of algorithms from which particular method is significantly better or significantly worse according to pair-wise comparisons by means of Friedman test coupled with Shaffer's post-hoc static procedure at $\alpha = 0.05$. The algorithm is called better/worse (**b** / **w**) than particular competitor when the difference between their averaged rankings is statistically significant (refer to Tables 12–17) and the algorithm considered (named in particular row) has lower/higher ranking (refer to Table 9).

| Allowed number of function calls | 5000 | | 20,000 | | 50,000 | | 100,000 | | 150,000 | | 500,000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **b** | **w** | **b** | **w** | **b** | **w** | **b** | **w** | **b** | **w** | **b** | **w** |
| AdapSS-JADE | 0 | 0 | 1 | 0 | 7 | 0 | 7 | 0 | 7 | 0 | 4 | 0 |
| ALC-PSO | 5 | 0 | 1 | 0 | 0 | 1 | 0 | 8 | 0 | 10 | 0 | 7 |
| AMALGAM | 3 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| AM-DEGL | 0 | 1 | 0 | 0 | 1 | 0 | 4 | 0 | 7 | 0 | 10 | 0 |
| ANS | 2 | 0 | 0 | 4 | 0 | 6 | 0 | 12 | 0 | 12 | 0 | 13 |
| ATPS-DE | 7 | 0 | 10 | 0 | 7 | 0 | 9 | 0 | 8 | 0 | 4 | 0 |
| CDE | 0 | 5 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 |
| CLPSO | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| CLPSO-DEGL | 7 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| CoBiDE | 0 | 4 | 0 | 0 | 1 | 0 | 11 | 0 | 13 | 0 | 10 | 0 |
| DE | 0 | 16 | 0 | 18 | 0 | 17 | 0 | 16 | 0 | 16 | 0 | 16 |
| DEGL | 0 | 4 | 0 | 2 | 0 | 1 | 0 | 3 | 0 | 3 | 0 | 4 |
| DE-SG | 0 | 9 | 0 | 5 | 0 | 6 | 0 | 8 | 0 | 4 | 0 | 0 |
| EPSDE | 2 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| FDSADE | 2 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| GA-MPC | 0 | 1 | 1 | 0 | 1 | 0 | 4 | 0 | 6 | 0 | 6 | 0 |
| GLPSO | 3 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| HCLPSO | 7 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| jDElscop | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 |
| LBBO | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| MDE-pBX | 3 | 0 | 10 | 0 | 7 | 0 | 7 | 0 | 6 | 0 | 1 | 0 |
| MPEDE | 0 | 1 | 0 | 0 | 1 | 0 | 8 | 0 | 10 | 0 | 9 | 0 |
| NMA | 0 | 1 | 0 | 4 | 0 | 6 | 0 | 9 | 0 | 5 | 0 | 1 |
| PMS | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 4 | 0 | 11 | 0 | 11 |
| PSO-iw | 13 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 0 | 7 | 0 | 11 |
| RA | 0 | 0 | 0 | 4 | 0 | 7 | 0 | 12 | 0 | 11 | 0 | 5 |
| Rcr-JADE | 0 | 0 | 3 | 0 | 12 | 0 | 11 | 0 | 11 | 0 | 7 | 0 |
| SADE | 2 | 0 | 6 | 0 | 7 | 0 | 5 | 0 | 5 | 0 | 4 | 0 |
| SapsDE | 2 | 0 | 7 | 0 | 7 | 0 | 7 | 0 | 2 | 0 | 2 | 0 |
| SFMDE | 2 | 0 | 0 | 3 | 0 | 6 | 0 | 12 | 0 | 11 | 0 | 7 |
| SPS-L-SHADE-EIG | 0 | 1 | 1 | 0 | 1 | 0 | 7 | 0 | 9 | 0 | 13 | 0 |
| SP-UCI | 0 | 16 | 0 | 10 | 0 | 7 | 0 | 2 | 0 | 2 | 0 | 0 |
| SspDE | 0 | 5 | 2 | 0 | 3 | 0 | 4 | 0 | 6 | 0 | 6 | 0 |

more runtime are not those that are especially highly ranked overall in our tests. Hence, the codes that are computationally demanding are not necessarily those that lead to the best results.

## 4. Conclusions

This study aims to search for the relationship between the number of allowed function calls and the relative performance of metaheuristics on numerical real-world problems, and to point out algorithms that are most suitable for particular computational budgets. The study is based on the testing of 33 metaheuristics (given in Table 2), proposed between 1960 and 2016, on 22 real-world problems from the CEC2011, each set with six computational budgets that differ by two orders of magnitude (from 5000 to 500,000 function calls). Each algorithm is run 30 times on each problem and computational budget; runs for different numbers of function calls are independent.

A strong relationship between the relative performance of metaheuristics and the allowed number of function calls is found. According to 22-problem averaged rankings, among the five algorithms that emerged as the best when the number of function calls is set to 5000, three are ranked within the worst 10 when the number of function calls is set to 500,000; the remaining ones (ATPS-DE [68] and HCLPSO [35]) are the 8th and 14th best respectively. All five of the best methods when the computational budget is set to 500,000 turn out to be among the poorest half of optimizers when computational budget is set to 5000. Moreover, the averaged rankings of the vast majority of algorithms (with five exceptions, out of 33 methods tested) noticeably vary with computational budget. Generally speaking, algorithms based on Particle Swarm Optimization and, to a lesser degree, novel types of metaheuristics like variants of Biogeography-based algorithms [51], Across Neighborhood Sampling [64] or Parallel Memetic Structures [6] achieve the best rankings when a low number of function calls is allowed. With some surprise it was found that the classical PSO with inertia weight proposed in 1998 [50] emerged as the best method from 33 competitors when the computational budget is set to only 5000 function calls. On the contrary, Genetic Algorithm or Differential Evolution methods, as well as simplex-based SP-UCI, perform relatively better when the number of function calls is the highest amongst those considered (500,000). This is especially highlighted for the adaptive,

**Table 19**
Example of comparison of executional runtime (in seconds) of all codes used in this study on selected problems (P1, P8 and P11.6) with maximum number of function calls set to 20,000. mean – mean runtime from 30 runs. st. dev. – standard deviation of the runtime from 30 runs. Codes that consumed more than 50% more time than the quickest ones are bolded.

| Executional runtime | P1 mean | P1 st.dev. | P8 mean | P8 st. dev. | P11.6 mean | P11.6 st. dev. |
|---|---|---|---|---|---|---|
| AdapSS-JADE | 2.075 | 0.027 | 4.850 | 0.132 | 2.235 | 0.040 |
| ALC-PSO | 1.861 | 0.018 | 4.523 | 0.022 | 2.046 | 0.008 |
| AMALGAM | **8.103** | 0.245 | **12.714** | 0.702 | **10.741** | 0.806 |
| AM-DEGL | 2.354 | 0.042 | 4.877 | 0.041 | 2.433 | 0.160 |
| ANS | **4.172** | 0.022 | **6.994** | 0.099 | **4.386** | 0.049 |
| ATPS-DE | 2.080 | 0.023 | 4.660 | 0.024 | 2.190 | 0.016 |
| CDE | 1.885 | 0.047 | 4.484 | 0.012 | 1.968 | 0.056 |
| CLPSO | **3.304** | 0.015 | 5.931 | 0.018 | **3.648** | 0.032 |
| CLPSO-DEGL | **2.906** | 0.032 | 5.662 | 0.031 | **4.284** | 0.031 |
| CoBiDE | 2.064 | 0.023 | 4.634 | 0.023 | 2.350 | 0.023 |
| DE | 1.980 | 0.011 | 4.630 | 0.063 | 1.976 | 0.010 |
| DEGL | 2.389 | 0.023 | 5.079 | 0.037 | **4.638** | 0.032 |
| DE-SG | 1.980 | 0.016 | 4.692 | 0.033 | 1.984 | 0.012 |
| EPSDE | 1.964 | 0.015 | 4.640 | 0.021 | 2.360 | 0.039 |
| FDSADE | 2.061 | 0.028 | 4.783 | 0.102 | 2.164 | 0.047 |
| GA-MPC | 2.460 | 0.027 | 4.842 | 0.054 | **3.698** | 0.026 |
| GLPSO | 2.171 | 0.031 | 4.788 | 0.021 | **3.794** | 0.008 |
| HCLPSO | 2.091 | 0.014 | 4.856 | 0.054 | 2.435 | 0.068 |
| jDElscop | 1.991 | 0.014 | 4.835 | 0.050 | 2.061 | 0.010 |
| LBBO | **9.237** | 1.280 | **14.059** | 0.298 | **2.897** | 0.183 |
| MDE-pBX | 2.166 | 0.021 | 4.791 | 0.027 | **3.719** | 0.150 |
| MPEDE | 2.264 | 0.035 | 4.884 | 0.061 | 2.001 | 0.032 |
| NMA | **3.728** | 0.038 | 5.476 | 0.037 | **4.815** | 0.035 |
| PMS | 2.632 | 0.045 | 4.583 | 0.028 | **5.545** | 0.102 |
| PSO-iw | 1.832 | 0.027 | 4.526 | 0.037 | 1.992 | 0.007 |
| RA | 1.938 | 0.024 | 4.650 | 0.022 | 1.927 | 0.016 |
| Rcr-JADE | 2.030 | 0.017 | 4.614 | 0.025 | 2.168 | 0.012 |
| SADE | 2.089 | 0.019 | 4.802 | 0.026 | 2.366 | 0.047 |
| SapsDE | 2.256 | 0.029 | 4.944 | 0.019 | 2.356 | 0.019 |
| SFMDE | 1.898 | 0.069 | 4.723 | 0.093 | 2.114 | 0.064 |
| SPS-L-SHADE-EIG | 2.106 | 0.017 | 4.741 | 0.013 | 2.284 | 0.013 |
| SP-UCI | **3.269** | 0.090 | 6.252 | 0.071 | **5.562** | 0.025 |
| SspDE | 1.919 | 0.017 | 4.535 | 0.024 | 2.222 | 0.018 |

distributed or composite variants, like SPS-L-SHADE-EIG [25], which is the top ranked for the highest computational budget, CoBiDE [61], AM-DEGL [43] or MPEDE [65]. Note that when 500,000 function calls are allowed, Genetic Algorithm GA-MPC [18], which was the winner of CEC2011 competition has been outperformed only by new approaches that were proposed after 2012.

Although with the increasing number of function calls the relative ranking of most algorithms either clearly improves or deteriorates, five algorithms perform similarly irrespective of the computational budget. However, these five methods either perform moderately or very poorly. For example, the basic DE algorithm is among two of the worst variants tested in each of the six competitions. On the other hand, the relation between the relative performance of some algorithms and the computational budget is more complicated. For example, a few Differential Evolution algorithms are ranked best with moderate computational budgets, and their relative rankings deteriorate when the number of allowed function calls is either increased or decreased. For historical Direct Search methods the reverse relationship is observed – namely they perform relatively better with both very low or very large computational budgets, than with moderate values.

It has been found that some algorithms, (especially SP-UCI [8], but also Rosenbrock's algorithm [48] and Linearized Biogeography-based Optimization [51]) are more specialized than others – in other words for fixed computational budget they perform very well on some real-world problems, but very poorly on the others. Generally, Particle Swarm Optimization, Differential Evolution or Genetic Algorithm variants tested do not show such large differences in performance on various types of problems.

It is found that the algorithms that win for the largest number of problems are often those based on Particle Swarm Optimization (when computational budget is low), or Differential Evolution or Genetic Algorithms (when computational budget is large). For specific computational budgets some Particle Swarm Optimization or Differential Evolution algorithms are ranked among the best five metaheuristics (out of 33 tested) on about 50% of problems, which shows their quite wide applicability to various numerical real-world problems, only if an adequate computational budget is set. Such methods include PSO-iw [50] and ALC-PSO [7] when only 5000 function calls are available, or SPS-L-SHADE-EIG [25], AM-DEGL [43] and CoBiDE [61] when 500,000 function calls are allowed. Algorithms that are the most frequent winners when the number of function calls is moderate or high are often those that use some methods of control parameters adaptation. When the num-

ber of function calls is high, one may also recommend the performance of approaches that modify coordinate system during run, including the relatively new SPS-L-SHADE-EIG [25] or CoBiDE [61], and the historical RA [48].

For almost all problems the differences between the performances of best algorithms are relatively low. On the contrary, one quarter of algorithms tested perform exceptionally poorer than the majority of methods for at least one problem and computational budget. This means that finding a single excellent method for particular problem is rare; on the other hand, some algorithms may be fully inappropriate for specific applications, even though they perform relatively well on many other problems.

An experiment was also conducted by correlating the ranks achieved by specific algorithm on various problems when computational budget is set to 5000 and 150,000. It was found that such correlation heavily depends on the algorithm considered. In the case of some Differential Evolution (CDE [4], MPEDE [65]) and Particle Swarm Optimization (ALC-PSO [7]) variants it may even be strongly negative, which means that such methods are, after a longer computational time, better ranked on problems on which they perform relatively poorly when computational budget is low.

It was also found that none of the five out of six algorithms tested that use variable population size (AMALGAM [59], ATPS-DE [68], SapsDE [60], jDElscop [3] and FDSADE [57]) was ranked among the worse one-third of methods according to 22-problems averaged rankings, in any out of six competitions. Although in some studies [40,45] it has already been pointed out that variable population size helps to avoid poor performance, the present finding may suggest that variable population size is helpful irrespective of the computational budget or various specific shortcomings of particular algorithms. In slight contradiction to this finding, the sixth tested algorithm with variable population size, SPS-L-SHADE-EIG [25], suffers from poor ranking when the computational budget is very low due to extremely large initial population size; however this feature coupled with linear population size reduction allows SPS-L-SHADE-EIG to become the best method when computational budget is large.

Although this was beyond the main goal of the paper, some partial tests of true computational runtime (measured in seconds) of specific codes used in this study have been performed on three selected problems, to verify whether codes that require more computational runtime lead to better results. We found that the true computational runtime of most codes is roughly similar, but some codes (unfortunately, often those obtained from their authors) require much more computational runtime than the majority. It is interesting to note that, taking into account just three real-world problems, the most runtime-greedy codes are not those that lead to the best results. We stress that this finding cannot be considered general, as it is based on very limited tests performed in a specific environment on a very few real-world problems.

Five main conclusions from the discussion given in this study are as follows:

1. None of the algorithms that perform best with the lowest considered computational budget are ranked highly when the number of allowed function calls is large. This confirms that exploitation-greedy algorithms, that often win in short-haul competition, cannot be efficiently used for more demanding problems when more function calls are available. Moreover, the reverse conclusion is also true – none of the algorithms that perform best on tests with a large number of allowed function calls are ranked well on tests with a very low computational budget.
2. The majority of algorithms perform relatively better with either a high, low, or moderate computational budget, hence most algorithms may be considered “specialized” with respect to the number of allowed function calls. Relative ranking of only a few, often poor methods was similar irrespective of the computational budget. For specific problems some algorithms may be among the best if the number of function calls is low, but are among the poorest if it is high, or vice versa. Hence claims that a particular method is useful for a particular problem without stating computational budget are doubtful. Some algorithms are fully unreliable on specific problems irrespective of computational budget, even though they may perform reasonably on other problems; hence an unfortunate choice of optimization method for a specific practical problem may lead to erroneous results.
3. Among tested algorithms, overall Particle Swarm Optimization variants often perform relatively better than the competitors when the number of allowed function calls is low. On the other hand, Genetic Algorithm and Differential Evolution methods perform relatively better when the computational budget is high.
4. Algorithms with variable population size are never among the worst in any presented comparisons. Although variable population sizes do not guarantee achieving good performance, it helps in avoiding pitfalls.
5. Algorithms that, with specific computational budget, perform very well according to ranks averaged over all considered problems are not necessarily those that win competitions for any specific application. Frequently the methods that perform best on some specific problems achieve poor results on others. Very good average ranks may be obtained by algorithms that perform well on vast majority of problems, even if they do not necessarily win for any specific one.

## Acknowledgments

This work was supported within statutory activities No 3841/E-41/S/2015 of the Ministry of Science and Higher Education of Poland. The publication has been partially financed from the funds of the Leading National Research Centre (KNOW) received by the Centre for Polar Studies for the period 2014-2018.

We would like to thank Prof. Ponnuthurai N. Suganthan for providing the code of the MDE_pBX, CLPSO and HCLPSO algorithms, Prof. Jasper A. Vrugt for providing the code of AMALGAM, and Dr. Wei Chu for providing the code of SP-UCI. We are also grateful to Prof. Ferrante Neri for his valuable comments on the SFMDE algorithm and to the authors of

[18,51,58,64,65] for making the codes of the ANS, GA-MPC, LBBO, MPEDE and SPS-L-SHADE-EIG algorithms and Shaffer's post-hoc statistical procedures widely available on the web.

## Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2016.12.028.

## References

[1] I. Boussaid, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117.
[2] J. Branke, J. Elomari, Meta-optimization for parameter tuning with a flexible computing budget, in: Proc. of 14th GECCO, Philadelphia, PA, USA, 2012, pp. 1245–1252.
[3] J. Brest, M.S. Maucec, Self-adaptive Differential Evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (11) (2011) 2157–2174.
[4] Z.H. Cai, W.Y. Gong, C.X. Ling, H. Zhang, A clustering-based Differential Evolution for global optimization, Appl. Soft Comput. 11 (1) (2011) 1363–1379.
[5] A. Caponio, F. Neri, V. Tirronen, Super-fit control adaptation in memetic Differential Evolution frameworks, Soft Comput. 13 (8-9) (2009) 811–831.
[6] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Inf. Sci. 227 (2013) 60–82.
[7] W.N. Chen, J. Zhang, Y. Lin, N. Chen, Z.H. Zhan, H.S.H. Chung, Y. Li, Y.H. Shi, Particle Swarm Optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.
[8] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high-dimensional problems, Inf. Sci. 181 (22) (2011) 4909–4927.
[9] M. Crepinsek, S.H. Liu, M. Mernik, Exploration and exploitation in Evolutionary Algorithms: a survey, ACM Comput. Surv. 45 (3) (2013) 1–33 Art. No. 35.
[10] M. Crepinsek, S.H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them, Appl. Soft Comput. 19 (2014) 161–170.
[11] S. Das, A. Abraham, U.K. Chakraboty, A. Konar, Differential Evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[12] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems, Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India, Dec. 2010.
[13] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 27–54.
[14] J. Derrac, S. Garcia, S. Hui, P.N. Suganthan, F. Herrera, Analyzing convergence performance of Evolutionary Algorithms: a statistical approach, Inf. Sci. 289 (2014) 41–58.
[15] S. Droste, T. Jansen, I. Wegener, Upper and lower bounds for randomized search heuristics in black-box optimization, Theory Comput. Syst. 39 (2006) 525–544.
[16] A.S. Dymond, A.P. Engelbrecht, S. Kok, P.S. Heyns, Tuning optimization algorithms under multiple objective function evaluation budgets, IEEE Trans. Evol. Comput. 19 (3) (2015) 341–358.
[17] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. 6th Int. Symp. Micromachine Human Sci., Nagoya, Japan, 1995, pp. 39–43.
[18] S.M. Elsayed, R.A. Sarker, D.L. Essam, GA with a new multi-parent crossover for constrained optimization, in: IEEE Congress on Evolutionary Computation, 2011, pp. 857–864.
[19] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, Evolving cognitive and social experience in particle Swarm Optimization through Differential Evolution: a hybrid approach, Inf. Sci. 216 (2012) 50–92.
[20] S. Garcia, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.
[21] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (5) (1986) 533–549.
[22] W.Y. Gong, A. Fialho, Z.H. Cai, H. Li, Adaptive strategy selection in Differential Evolution for numerical optimization: an empirical study, Inf. Sci. 181 (24) (2011) 5364–5386.
[23] W.Y. Gong, Z.H. Cai, Y. Wang, Repairing the crossover rate in adaptive Differential Evolution, Appl. Soft Comput. 15 (2014) 149–168.
[24] Y.J. Gong, J.J. Li, Y. Zhou, Y. Li, H.S.H. Chung, Y.H. Shi, J. Zhang, Genetic Learning Particle Swarm Optimization, IEEE Trans. Cybern. 46 (10) (2016) 2277–2290.
[25] S.M. Guo, J.S.H. Tsai, C.C. Yang, P.H. Hsu, A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set, in: Proc. IEEE Congress on Evolutionary Computation, Sendai, Japan, 2015, pp. 1003–1010.
[26] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proc. IEEE Int. Conf. Evol. Comput., Nagoya, Japan, 1996, pp. 312–317.
[27] N. Hansen, A. Auger, R. Ros, S. Finck, P. Posik, Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009, in: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2010, New York, ACM, 2010, pp. 1689–1696.
[28] N. Hansen, A. Auger, S. Finck, R. Ros, Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup. INRIA research report RR-7215, 2010.
[29] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive Differential Evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst. Man Cybern. Part B – Cybernetics 42 (2) (2012) 482–500.
[30] Y. Jin, Surrogate-assisted evolutionary computation: Recent advances and future challenges, Swarm Evol. Comput. 1 (2011) 61–70.
[31] G. Karafotias, M. Hoogendoorn, A.E. Eiben, Parameter control in Evolutionary Algorithms: trends and challenges, IEEE Trans. Evol. Comput. 19 (2) (2015) 167–187.
[32] T.G. Kolda, R.M. Lewis, V. Torczon, Optimization by direct search: new perspectives on some classical and modern methods, SIAM Rev. 45 (3) (2003) 385–482.
[33] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, Convergence properties of the Nelder–Mead simplex method in low dimensions, SIAM J. Optim. 9 (1) (1998) 112–147.
[34] J.J. Liang, A.K. Qin, P.N. Suganthan, S Baskar, Comprehensive Learning Particle Swarm Optimizer for global opti-mization of multimodal functions, IEEE Trans. Evol. Comput. 10 (3) (2006) 281–295.
[35] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning Particle Swarm Optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.
[36] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential Evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.
[37] M. Mernik, S.H. Liu, D. Karaboga, M. Crepinsek, On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation, Inf. Sci. 291 (2015) 115–127.
[38] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, C. Weihs, Analyzing the BBOB results by means of benchmarking concepts, Evol. Comput. 23 (1) (2015) 161–185.

[39] J.A. Nelder, R. Mead, A simplex-method for function minimization, Comput. J. 7 (4) (1965) 308–313.
[40] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.
[41] Q.K. Pan, P.N. Suganthan, L. Wang, L. Gao, R. Mallipeddi, A differential evolution algorithm with self-adapting strategy and control parameters, Comput. Oper. Res. 38 (1) (2011) 394–408.
[42] A.P. Piotrowski, J.J. Napiorkowski, A. Kiczko, Differential Evolution algorithm with separated groups for multi-dimensional optimization problems, Eur. J. Oper. Res. 216 (2012) 33–46.
[43] A.P. Piotrowski, Adaptive Memetic Differential Evolution with global and local neighborhood-based mutation operators, Inf. Sci. 241 (2013) 164–194.
[44] A.P. Piotrowski, Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions, Inf. Sci. 297 (2015) 191–201.
[45] A.P. Piotrowski, Review of Differential Evolution population size, Swarm and Evolutionary, Computation (2017), doi:10.1016/j.swevo.2016.05.003.
[46] P. Posik, W. Huyer, L. Pal, A comparison of global search algorithms for continuous black box optimization, Evol. Comput. 20 (4) (2012) 509–541.
[47] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential Evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[48] H.H. Rosenbrock, An automated method for finding the greatest or least value of a function, Comput. J. 3 (3) (1960) 175–184.
[49] J.P. Shaffer, Modified sequentially rejective multiple test procedures, J. Am. Stat. Assoc. 81 (395) (1986) 826–831.
[50] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceeding in IEEE Congress on Evolutionary Computation (CEC), 1998, pp. 69–73.
[51] D. Simon, M.G.H. Omran, M. Clerc, Linearized biogeography-based optimization with re-initialization and local search, Inf. Sci. 267 (2014) 140–157.
[52] K. Sorensen, Metaheuristics—the metaphor exposed, Int. Trans. Oper. Res. 22 (2015) 3–18.
[53] G. Squillero, A. Tonda, Divergence of character and premature convergence: a survey of methodologies for promoting diversity in Evolutionary Optimization, Inf. Sci. 329 (2016) 782–799.
[54] R. Storn, K.V. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[55] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for Differential Evolution, in: Proc. IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013, pp. 71–78.
[56] R. Tanabe, A. Fukunaga, Tuning Differential Evolution for cheap, medium, and expensive computational budgets, in: Proc of the IEEE Congress on Evolutionary Computation, Sendai, Japan, 2015, pp. 2018–2025.
[57] V. Tirronen, F. Neri, Differential Evolution with fitness diversity self-adaptation, in: Nature-Inspired Algorithms for Optimisation, SCI 193, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 199–234.
[58] A. Ulas, O.T. Yildiz, E. Alpaydin, Cost-conscious comparison of supervised learning algorithms over multiple data sets, Pattern Recognit. 45 (2012) 1772–1781.
[59] J.A. Vrugt, B.A. Robinson, J.M. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Trans. Evol. Comput. 13 (2) (2009) 243–259.
[60] X. Wang, S.G. Zhao, Differential Evolution algorithm with self-adaptive population resizing mechanism, Math. Prob. Eng. (2013) Art. No. 419372, doi:10.1155/2013/419372.
[61] Y. Wang, H.X. Li, T.W. Huang, L. Li, Differential Evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[62] T. Weise, R. Chiong, J. Lassig, K. Tang, S. Tsutsui, W.X. Chen, Z. Michalewicz, X. Yao, Benchmarking optimization algorithms: an open source framework for the traveling salesman problem, IEEE Comput. Intell. Mag. 9 (3) (2014) 40–52.
[63] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[64] G.H. Wu, Across neighborhood search for numerical optimization, Inf. Sci. 329 (2016) 597–618.
[65] G.H. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H.K. Chen, Differential Evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[66] A. Zamuda, J. Brest, Self-adaptive control parameters' randomization frequency and propagations in Differential Evolution, Swarm Evol. Comput. 25 (2015) 72–99.
[67] J. Zhang, Z.C. Sanderson, JADE: adaptive Differential Evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[68] W. Zhu, Y. Tang, J.A. Fang, W.B. Zhang, Adaptive population tuning scheme for Differential Evolution, Inf. Sci. 223 (2013) 164–191.