Distributed Evolutionary Algorithms with Hierarchical Evaluation

Ioannis Kampolis, Kyriakos C. Giannakoglou

▶ **To cite this version:**

Ioannis Kampolis, Kyriakos C. Giannakoglou. Distributed Evolutionary Algorithms with Hierarchical Evaluation. Engineering Optimization, 2009, 41 (11), pp.1037-1049. 10.1080/03052150902890072 . hal-00545365

HAL Id: hal-00545365
https://hal.science/hal-00545365

Submitted on 10 Dec 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed Evolutionary Algorithms with Hierarchical Evaluation

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2008-0219.R3
Manuscript Type:	Original Article
Date Submitted by the Author:	09-Mar-2009
Complete List of Authors:	Kampolis, Ioannis; National Technical University of Athens Giannakoglou, Kyriakos; National technical University of Athens
Keywords:	Design Optimization, Evolutionary Algorithms, Metamodels, Hierarchical Search, Distributed Search

Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.

kampolis.tex
gENO2e.cls
gENO.bst
mkkalgo.sty
amsthm.sty
enumlist.sty
kampolis.bib
kampolis_aux.tex
figs-tables.tex

scholarONE™
Manuscript Central

For Peer Review Only

RESEARCH ARTICLE

Distributed Evolutionary Algorithms with Hierarchical Evaluation

Ioannis C. Kampolis and Kyriakos C. Giannakoglou*

National Technical University of Athens,
Lab. of Thermal Turbomachines,
Parallel CFD & Optimization Unit,
P.O. Box 64069, Athens 157 10, GREECE,
e-mail: kgianna@central.ntua.gr

(*Received 00 Month 200x; final version received 00 Month 200x*)

A distributed evolutionary algorithm based on a hierarchy of (fitness or cost function) evaluation passes within each deme, efficient in solving engineering optimization problems is presented. Starting with a non–problem–specific evaluations (using surrogate models or metamodels, trained on previously evaluated individuals) and ending up with high–fidelity problem–specific evaluations, intermediate passes rely on other available lower fidelity problem–specific evaluations with lower CPU cost per evaluation. The sequential use of evaluation models or metamodels, of different computational cost and modeling accuracy, by screening the generation members to get rid of non–promising individuals, leads to reduced overall computational cost. The distributed scheme is based on loosely coupled demes that exchange regularly their best–so–far individuals. Emphasis is put on the optimal way of coupling distributed and hierarchical search methods. The proposed method is tested on mathematical and compressor cascade airfoil design problems.

Keywords: Design Optimization, Evolutionary Algorithms, Metamodels, Hierarchical search, Distributed Search

*Corresponding author

ISSN: 0305-215X print/ISSN 1029-0273 online
© 200x Taylor & Francis
DOI: 10.1080/0305215YYxxxxxxx
http://www.informaworld.com

For Peer Review Only
URL: http://mc.manuscriptcentral.com/geno Email: A.B.Templeman@liverpool.ac.uk

1. Introduction

Evolutionary algorithms (EAs) were quickly adopted to solve engineering design or optimization problems due to their ability to handle more than one objectives and their easy coupling with any commercial or in–house analysis software. The literature is full of engineering problems solved by means of EAs (genetic algorithms, evolution strategies or, less frequently, evolutionary programming). The major drawback of EA–based optimization is the necessity of carrying out high numbers of calculation of the fitness of cost function values (to be referred to as "evaluations") before reaching the optimal solution(s). This is, in fact, a common feature of all stochastic, population–based search algorithms. From this viewpoint, the hierarchical and distributed schemes proposed in this paper are not limited to EAs but can be extended to any other population–based method. Taking this into consideration, herein a generalized $(\mu, \lambda)EA$ (with μ parents and λ offspring) implementing standard evolution operators, will be used as the core search engine.

Structuring the evolutionary search in hierarchical manner is a means to reduce the overall CPU cost for carrying out the optimization, combining different tools (Kampolis *et al.* 2007, Kampolis and Giannakoglou 2008). The gain from using hierarchical search is superimposed to that expected from the use of a "better" EA, "better" evolution operators and so forth. According to the literature survey, existing hierarchical schemes can be classified to algorithms relying on different evaluation methods to computer the fitness or cost function of candidate solutions (with different fidelity and CPU cost) (Eby *et al.* 1998, Herrera *et al.* 1999, Sefrioui and Périaux 2000, Karakasis *et al.* 2007, Kampolis *et al.* 2007, Kampolis and Giannakoglou 2008), different search techniques (Muyl *et al.* 2004, Poloni *et al.* 2000, Knowles and Corne 2000) and different chromosome sizes (Lin *et al.* 1994) or numbers of design variables (Désidéri and Janka 2003, Duvigneau *et al.* 2006). The present paper is concerned with the first class of hierarchical methods, i.e. those relying on various evaluation tools (problem– or non–problem–specific ones, as it will be explained in section 2) and focuses on implementation issues. Some relevant hierarchical schemes are overviewed below.

In (Eby *et al.* 1998), EAs are organized in levels each of which evolves chromosomes of different size, using different evaluation methods. Migrations are directed from the lower level (the one associated with the low–cost, low–fidelity evaluation model and coarse chromosomes) to the higher one (with the high–fidelity, costly evaluation model and fine chromosomes). In (Herrera *et al.* 1999), the entire EA population splits into demes in which evolution is differently tuned, giving rise to demes with enhanced exploration capabilities and others which try to exploit the previously collected data. The exchange of promising solutions is carried out via a number of migration schemes that depend upon the connectivity of demes and their orientation (exploration/exploitation oriented). A grouping in levels is adopted according to whether each deme is either exploration– or exploitation–oriented, however all demes make use of the same evaluation software. In (Sefrioui and Périaux 2000), a binary tree topology splitting the search into three levels associated with different evaluation models is used. The most accurate and costly software is assigned to the highest level, configured to promote exploitation. The lowest level is fully exploration oriented, using high mutation probability. One third of the population of each level is allowed to migrate to its upper level and these correspond to the best–so–far solutions. An equal number of randomly selected individuals are allowed to migrate downwards. All immigrants are re–evaluated using the destination level model.

In (Karakasis *et al.* 2007, Kampolis *et al.* 2007, Kampolis and Giannakoglou 2008),

For Peer Review Only

the distributed search of optimal solutions is structured in levels, each of which employs an EA relying on surrogate evaluation models, i.e. the so–called metamodels. However, there are significant differences between these three works and the present one. The former deal with hierarchical (or multilevel) optimization algorithms where each level is supported by its own single– or multi–population EA or, depending on the configuration, a deterministic search method, see (Kampolis and Giannakoglou 2008). To make it as fast as possible, all EAs are assisted by surrogate models (metamodels) which are trained locally on previously evaluated (on the level's model) individuals. Note that, in the aforementioned three papers: (a) the number of demes per level may be different, (b) migration occurs between successive levels by neglecting their structure in demes, (c) each level maintains its own database of previously evaluated individuals which is used for training its own metamodels, (d) one– or two–way inter–level and various intra–level (inter–deme) migration schemes are used and (e) none of the levels is associated with a metamodel only.

To contrast methods presented in (Karakasis *et al.* 2007, Kampolis *et al.* 2007, Kampolis and Giannakoglou 2008) with the present paper, the former will be referred to as *hierarchical distributed metamodel–assisted EAs* whereas the newly proposed method is a *distributed hierarchical EA* in which properly trained metamodels are used for the evaluation of candidate solutions during the lower pass. In the present method, the hierarchical search is carried out within each deme (this is why this is called *distributed hierarchical*, rather than the other way round). As already stated, the lower pass is carried out by metamodels. These are locally valid, trained separately for each new offspring using the closest, previously evaluated individuals. The use of metamodels starts upon completion of the first few generations, i.e. after collecting a minimum number of training patterns. Moving upwards, problem–specific evaluation tools of increasing fidelity and CPU cost are employed. In the upward direction, the number of evaluations is reduced by promoting only a few best performing individuals. The inter–deme communication occurs regularly by exchanging their best–so–far individuals.

In section 3, two mathematical and two aerodynamic shape optimization problems are used to demonstrate the efficiency of the proposed method. In all examined cases, the hierarchical search is built using one metamodel (E_0) and two problem–specific evaluation models (low–fidelity E_1 and high-fidelity E_2). It should be clearly stated that E_1 and E_2 may rely on different physical assumptions or numerical settings. For instance, in aerodynamic shape optimization problems, E_1 could be a simplified flow model running on a coarse mesh to reduce the CPU cost and E_2 a much more expensive flow solver which, on an appropriately fine mesh, yields the desired modeling accuracy; alternatively, E_1 and E_2 may use the same flow model with relaxed convergence criteria. The latter was used in both aerodynamic shape optimization problems, see sections 3.3 and 3.4. Regarding the mathematical problems analyzed in sections 3.1 and 3.2, in the absence of real low–fidelity models, the latter were artificially devised by perturbing (i.e. by introducing noise to) the real equations and by considering that the CPU cost ratio is the same as in the aerodynamic design problems.

2. Distributed Hierarchical Evolutionary Algorithm

The proposed algorithm is built on the basis of a distributed EA (DEA), (Karakasis and Giannakoglou 2003, Doorly *et al.* 1999, Herrera and Lozano 2000). In $DEAs$, the evolution takes place simultaneously on a small number of medium–sized sub–populations,

For Peer Review Only

the so–called islands or demes. The use of a DEA, rather than a single–population EA, is more efficient in terms of CPU cost. This is demonstrated in section 3, where single–population and distributed algorithmic variants are compared. The evolution operators are applied within each deme, though each deme can be associated with different evolution policies (crossover scheme, mutation probability, elitism, etc) promoting either exploration or exploitation (Doorly and Peiró 1997). An inter–deme migration operator is employed by exchanging the best performing (and/or some random) individuals; different variants of $DEAs$ can be devised by changing the inter–deme communication topology (ring, grid, etc.), the inter–deme migration frequency and rate as well as the selection and replacement policies, (Alba and Tomassini 2002).

In this paragraph, the hierarchical structure is ignored so as to go through the properties of the distributed search scheme. Under this assumption, all demes are initially assigned the same evolution parameters. After a user–defined number of generations, deme(s) containing the best–so–far solution update their evolution parameters to exploit and evolve further this solution whereas the rest become more exploration–oriented by increasing the mutation probability. Over and above, taboo regions around the best–so–far solutions are introduced in the exploration–oriented demes to direct search towards unexplored regions in the design space. Whenever a new offspring falls into a taboo region, its fitness is penalized. Taboo regions adapt themselves dynamically to newly appearing optimal solutions. A single database serves to store all previously evaluated individuals from all demes, since there is a single hierarchical level with a single evaluation tool.

Hierarchy is employed within each deme. In what follows, both single– and multi–objective minimization problems are considered. In the proposed method (fig. 1), different evaluation tools (either problem–specific models, such as Computational Fluid Dynamics - CFD - software tools of different accuracy and CPU cost or surrogate models) are used to evaluate the offspring population(s), according to the proposed hierarchical scheme. Let $\mathcal{P}_\lambda^g$ denote the set of λ offspring of a deme at the g-th generation and $E_s,\ (s = 0, ..., S)$ the $S+1$ available evaluation tools. By convention, E_S and E_0 correspond to the most expensive and the cheapest evaluation software, respectively. It should become clear that all evaluation models use the same number of design variables. Schemes that use different (coarse and fine) parameterizations on each level can alternatively be used, as shown elsewhere (Kampolis and Giannakoglou 2008), but are beyond the scope of this paper.

Symbol λ_s denotes the maximum number of offspring per generations and deme to be evaluated on E_s; this is computed as $\lambda_s = \lambda \prod_{i=0}^{s} w_i$. The user–defined parameters $w_i \in [0, 1]$ stand for the percentage of individuals evaluated by E_{i-1} to will undergo re–evaluation on E_i. It is evident that $w_0 = 1$ (so $\lambda_0 \equiv \lambda$) whereas w_i decreases with i. In this manner, E_{i+1} is expected to evaluate only a subset of the individuals previously evaluated on E_i.

In aerodynamic shape optimization problems, unrealistic shapes often come up during the first generations. The numerical solution of the flow equations around or inside these shapes, through the CFD tool, may face serious convergence problems which reflect the complex flow physics or limitations of the solution model. In such a case, the presence of failed evaluations is handled as follows: the pool of offspring selected for evaluation using E_i is initially restricted to those evaluated on E_{i-1}; if these are exhausted, individuals evaluated using $E_{i-2}, E_{i-3}, ..., E_0$ are successively considered.

The lowest pass evaluation is based on the non–problem–specific surrogate model E_0, i.e. the metamodel. Metamodels (Giannakoglou 2002, Jin 2005, Karakasis and Giannakoglou 2005, Lim *et al.* 2008, Zhou *et al.* 2007, Emmerich *et al.* 2006) are generic

For Peer Review Only

interpolation or approximation methods such as polynomial regression, artificial neural networks, etc, which, after being trained on previously seen solutions, are used in place of E_1 to screen out non–promising candidate solutions at very low CPU cost. In this manner, only a few of the best offspring undergo evaluations on $E_i,\ i>0$. The evolution operators are applied on individuals evaluated on either the metamodel or problem–specific models of any fidelity. In this paper, without loss in generality, radial basis function (RBF) networks (Haykin 1999) are used as metamodels. For efficient training algorithms of metamodels during the evolution of a $MAEA$ the reader should refer to (Karakasis and Giannakoglou 2005). Training the metamodels requires a pool (or database) of samples, shared among all demes. So, evaluations on E_0 are postponed until the database of previously evaluated individuals (using E_1, considered this to be the lowest–fidelity problem–specific tool) exceeds a user–defined minimum number of entries. Upon completion of this starting phase (during which $w_1=1$) a metamodel is separately trained for each new offspring, using neighboring patterns from the database. Each metamodel is used to approximate the objective vector values, i.e. to compute $\tilde{\mathbf{F}}(\mathbf{x})$. Individuals evaluated on metamodels are assigned $s=0$.

The proposed hierarchical EA, as used within each deme, is described below in detail:

Algorithm DHEA (Distributed Hierarchical EA).

DHEA1. [Start] The newly created offspring population $\mathcal{P}_\lambda^g$ (g is the generation counter) must be evaluated. The evaluation software counter is set to $s=0$.

DHEA2. [Non–problem–specific evaluations] If a "sufficient" (user–defined) number of entries in the database exist, pre–evaluate all offspring using properly trained metamodels, otherwise $w_1=1$. Set $s=1$.

DHEA3. [Problem–specific evaluations] Instead of evaluating the entire offspring population on a single evaluation software, the following actions are taken:

DHEA3a. [Screening] Populate list $\mathcal{L}$ with the λ_s best individuals, according to the preceding evaluation pass.

DHEA3b. [Evaluation] Evaluate all $\mathcal{L}$ members on E_s.

DHEA3c. [Utility assignment] A scalar cost function $\phi(\mathbf{x})$ is calculated based on the objective values of each individual $\mathbf{x}$, $\mathbf{F}(\mathbf{x})$ ($\mathbf{F}\in\mathcal{R}^M$). Computing $\phi(\mathbf{x})$ depends upon the number of objectives: in single–objective problems (SOO) $\phi(\mathbf{x}) = F_1(\mathbf{x})$, whereas in multi–objective problems (MOO, (Coello Coello *et al.* 2002, Deb 2001)) techniques such as SPEA (Zitzler *et al.* 2002) & NSGA (Srinivas and Deb 1995, Deb *et al.* 2002), etc are employed. It should be noted that individuals evaluated on E_{s_1} are allowed to dominate the ones evaluated on E_{s_2} provided that $s_1\geq s_2$. So $\mathbf{x}^{(p)}$ dominates $\mathbf{x}^{(q)}$ ($\mathbf{x}^{(p)}\prec\mathbf{x}^{(q)}$) if and only if

$$\forall m\in(1,...,M): F_m^{(p)}\leq F_m^{(q)}\wedge\exists m: F_m^{(p)}<F_m^{(q)}\wedge\mathcal{S}(\mathbf{x}^{(p)})\geq\mathcal{S}(\mathbf{x}^{(q)})$$

where $\mathcal{S}(\mathbf{x}^{(p)})$ denotes the software number used to evaluate $\mathbf{x}^{(p)}$.

DHEA3d. [Next pass] If $s<S\Rightarrow s\leftarrow s+1$ and go to step DHEA3a.

DHEA4. [Migration & Sharing] Apply the migration operator. Redefine the taboo regions and penalize offspring and parent populations.

DHEA5. [Evolutionary Operators] Generate the offspring population $\mathcal{P}_\lambda^{g+1}$ through the evolution operators.

DHEA6. [Termination] Unless a termination criterion is met, set $g\leftarrow g+1$ and return to step DHEA1. ∎

For Peer Review Only

The end of step DHEA3b introduces a synchronization barrier by suspending the evolution until all evaluation jobs on E_s be completed. On a multiprocessor platform, this may cause a significant decrease in the overall parallel efficiency of the algorithm if λ_s is less than the number of available processors. The distributed variant of the algorithm is used as a remedy: the aforementioned algorithm is applied to each deme separately. The concurrent evolution of all demes aims to keep all computational resources busy, since each deme may evolve faster or slower than the others and the cost per evaluation is not necessarily fixed.

3. Applications

Two function minimization problems are firstly presented, followed by two aerodynamic shape optimization problems.

The mathematical problems (sphere and Ackley functions) are very fast to run and their solution was repeated many times, using different pseudo–random number generator (PRNG) seed states. In the CFD–based optimization problems, a viscous–inviscid flow interaction method (based on the coupling of an integral boundary layer solver with an Euler equations solution method; code MISES, (Drela and Giles 1987)) served as the problem–specific evaluation software (E_2). Apart from the metamodel (E_0) and the high–fidelity tool (E_2), the same flow solver with relaxed convergence criteria was employed as the low–fidelity problem–specific tool (E_1), with a significantly lower CPU cost (average cost ratio of E_1 and E_2 equal to $1:10$). There are two reasons for choosing MISES as evaluation tool on two hierarchical passes: (a) it is an adequate tool for the analysis of the cascade flows under consideration and (b) it is very fast, since the CPU cost per evaluation is around 20 seconds on a commodity workstation. Any other CFD software (such as a Navier-Stokes solver with any turbulence model) could be used instead, by increasing however the CPU cost per evaluation along with the overall optimization cost.

All problems are solved using five algorithms:

a. a conventional single–population EA,
b. a single–population, two–pass hierarchical EA using the two problem–specific models, i.e. E_1 and E_2 (HEA),
c. its distributed variant ($DHEA$),
d. a single–population, three–pass hierarchical EA, using the metamodel E_0 and the same two problem–specific models ($HEAm$),
e. its distributed variant ($DHEAm$).

3.1. *Sphere function minimization*

The sphere function (Suganthan *et al.* 2005), with $N=30$ degrees of freedom ($\mathbf{x}\in\mathcal{R}^N$) is defined as:

$$f_2(\mathbf{x}) = (\mathbf{x}-\mathbf{o})(\mathbf{x}-\mathbf{o}) + f_{bias} \tag{1}$$

where $f_{bias}=-450$ and $\mathbf{o}\in\mathcal{R}^N$ is an offset vector, ($\mathbf{o}$ values can be downloaded from http://www3.ntu.edu.sg/home/EPNSugan/). In this study, eq. 1 served as the high–fidelity

For Peer Review Only

model (E_2). The low fidelity model (E_1) was "artificially" constructed as

$$f_1(\mathbf{x}) = f_2(\mathbf{x}) + 0.1 \sum_{i=1}^{N} x cos\,(x) \tag{2}$$

A hypothetical CPU cost ratio ($E_2 : E_1$) of $10:1$ was assigned to the two models to simulate the CPU cost ratio of the CFD tools used in the compressor design problems presented in sections 3.3 and 3.4 (see comments in the introductory paragraph of this section). In all algorithmic variants used to solve the same problem, the (μ, λ) EA was configured with $\lambda = 60$ offspring and $\mu = 30$ parents. Gray binary coding was used. For the distributed schemes (algorithms c and e) the population has split into two demes (2×30); the migration operator was applied every two generations, allowing five immigrants per deme to replace the worst performing offspring. A minimum of 350 evaluations were needed before enabling the metamodels in algorithms d and e. In the variants that employed metamodels, $\lambda_1 = 20$ (corresponds to $w_1 = 1/3$) and $\lambda_2 = 2$ (or $w_2 = 0.10$) individuals were qualified for evaluation on E_1 and E_2, respectively. Table 1 displays the performance of the five algorithms tested for a total CPU cost of 2000 units. One cost unit was assigned to each evaluation using E_2. Each algorithm was repeated 25 times using different PRNG seed states. The last row of the table presents the t-test value that compares the efficiency of two successive algorithms. The threshold (minimum) value for t_0 that ensures a significantly better performance is $t_{0,thres} = 2.3926$; it is easily concluded that $DHEAm$ (last column) is significantly better than the other algorithms tested. The convergence histories of the best–so–far solutions for all algorithms are plotted in fig. 3.

3.2. *Ackley function minimization*

The multimodal Ackley function to be minimized, (eq. 3) with $\alpha = 20, b = 0.2, c = 2\pi, \delta x = 0.0$ and $-32.768 \leq x_i \leq 32.768$ is defined as

$$f(\mathbf{x}) = -\alpha \exp\left(-b\sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}}\right) - \exp\left(\frac{\sum_{i=1}^{N} \cos(c(x_i - \delta x))}{N}\right) + \alpha + \exp(1) \tag{3}$$

with $N = 30$ degrees of freedom. Likely the sphere's case, herein also a "low–fidelity" model was artificially devised based on eq. 3 (i.e. the high–fidelity model, or E_2). The E_1 model was obtained by changing the original parameters to $\alpha = 18, b = 0.15, c = 1.8\pi, \delta x = 0.3$. As previously, the two models were assigned a hypothetical CPU cost ratio $10:1$ to conform with the CPU cost ratio of the CFD models used in the compressor design problem presented in sections 3.3 and 3.4. So, one cost unit was assigned to each evaluation on E_2 (eq. 3) and 0.1 cost units to each evaluation on E_1.

In all algorithmic variants a $(40, 80)$ EA configuration with gray binary coding was used. For the distributed schemes (algorithms c and e) the population was split into two demes (2×40); the migration policy was the same with the previous case; so, five well performing individuals per deme migrated every two generations and replaced the worst performing offspring in the destination deme. In this case, $w_2 = 0.05$, allowing $\lambda_2 = 4$ evaluations on the high–fidelity model at each generation. In algorithms d and e employing metamodels, a minimum number of 300 evaluations on the low–fidelity model E_1 had to be carried out before enabling metamodels; then, $\lambda_1 = 40$ (on E_1) and

For Peer Review Only

$\lambda_2 = 2$ (on E_2) evaluations were carried out. Table 2 displays the performance of the five algorithms tested for a total CPU cost of 1000, 5000 and 10000 cost units; each algorithm was repeated 25 times using different PRNG seed states. Fig. 4 displays the average convergence of each algorithm at the cost of 1000 and 10000 cost units and shows that the conventional EA was outperformed by all hierarchical variants. Also, each distributed scheme performed faster than the corresponding single–population one. The three–pass algorithms (d and e, using metamodels and the two problem–specific models) were able to capture a better solution than the corresponding two–pass (b and c, using only the problem–specific models) at the same CPU cost, with significantly better convergence characteristics as shown by the high t_0 values.

3.3. *Single–objective compressor blade airfoil design*

This problem is related to the design of a compressor cascade airfoil for minimum total pressure losses (i.e. for minimum loss coefficient ω). The flow conditions are $M_{2,is} = 0.45, \alpha_1 = 47^o, Re = 8.41 \cdot 10^5$. The airfoil shape was parameterized using two Bézier curves, one for each side (pressure, suction), with 14 degrees of freedom in total. Constraints were imposed on the minimum airfoil thickness T at various chordwise positions (C is the chord length) and the minimum flow turning:

$$T(0.3C) \geq 0.08C, \; T(0.6C) \geq 0.06C, \; T(0.9C) \geq 0.01C,$$
$$\alpha_1 - \alpha_2 \geq 19^o$$

In all five algorithms tested, the offspring and parent population were $\lambda = 60$ and $\mu = 15$. Gray binary coding was used. For the distributed algorithms (c and e) the offspring population was split into three demes (3×20); the inter–deme migration operator was applied every eight generations, allowing two immigrants per deme to replace the worst performing individuals. The low–fidelity problem–specific software E_1 was used to evaluate the entire population and only the 10% percent of its best performing offspring were re–evaluated on E_2. In algorithms involving metamodels (cases d and e), a minimum number of 300 training patterns were recorded in the database from evaluations on E_1, before enabling the use of metamodels. Then, $\lambda_1 = 4$ offspring were evaluated on E_1; only the best one among them was re–evaluated on E_2 ($\lambda_2 = 1$).

Fig. 5 illustrates the performance of the five algorithms. The total CPU cost of each optimization, measured in terms of cost units, is shown on the x-axis. One cost unit is equal to the CPU cost of one E_2 evaluation. Each call to E_1 costs as many as 0.1 CPU cost units. The algorithms were allowed to run for up to 600 CPU cost units. All hierarchical variants outperformed the conventional EA. The distributed variants outperform the corresponding single–population ones and the performance increased further if metamodels were used. Fig. 6 illustrates the pressure coefficient c_P distribution of the optimal blade along with its shape, as obtained by the $DHEAm$ which led to a cascade airfoil with $\omega = 0.0213$.

3.4. *Two–objective compressor blade airfoil design*

The last case is related to a similar design as the one presented in section 3.3. Two targets were imposed aiming at the minimization of total pressure losses (i.e. the ω coefficient) and the maximization of the static pressure rise $\frac{p_2}{p_1}$, where indices 1, 2 denote the cascade

For Peer Review Only

inlet and outlet, respectively. The constraint on the flow turning (see previous case) was not imposed since this was practically included in the second objective; in contrast, constraints on the minimum thickness were imposed, as in the previous case.

In this problem, a different configuration was used since, in MOO, search does not focus on a single region in the design space. Setting $w_2 = 0.1$ yields $\lambda_2 = 6$ evaluations on E_2 per generation; in the algorithms which also employ metamodels, $\lambda_1 = 30$ and $\lambda_2 = 3$. All the other settings were as in section 3.3.

As in the previous cases, the same five schemes were used to solve this problem, seeking for approximations to the Pareto front of non–dominated solutions. In order to compare their efficiencies, the hypervolume indicator (Zitzler *et al.* 2007) is plotted in fig. 7; the latter quantifies the part of the non–dimensional objective space (up to a user–defined reference point) dominated by the front. Fig. 7 illustrates results using the NSGA–2 (Deb *et al.* 2002) fitness assignment technique. Similar conclusions to the previous case can be drawn; the hierarchical use of the evaluation software significantly accelerates the convergence and a better front is obtained. The front of non–dominated solutions obtained by the $DHEAm$ algorithm is presented in fig. 8.

4. Conclusions

A distributed hierarchical optimization method, based on a series of problem–specific evaluation models and locally valid metamodels, trained on the fly during the evolution, was presented. It was proved that, in all cases, the optimization variant denoted by $DHEAm$ outperforms any other search method based on a different implementation of the same "ingredients". An important finding is that, according to the presented cases, the EA is significantly accelerated when the hierarchical evaluation technique is employed. Furthermore, its distributed variant (i.e. by using hierarchical evaluation inside each deme, employing taboo regions and allowing the demes to regularly exchange promising individuals) leads to higher quality of the final solution as well as higher parallel efficiency of the algorithm. The additional use of local metamodels (trained on a small subset of previously evaluated candidates) as the least accurate, and thus cheap, evaluation tool in the hierarchy is proved to further reduce the overall CPU cost. It should be noted that the proposed method is not restricted to the evolutionary algorithm used herein but can be extended to any other "rival" stochastic search technique, such as evolution strategies with covariant matrix adaptation (Auger and Hansen 2005) or particle swarm optimization (Langdo and Poli 2005), etc. Conclusions on the optimal combination of hierarchical and distributed search schemes are expected to be the same. Also, the RBF networks used as metamodels in this paper could be replaced by any other artificial neural network, response surface method, the kriging metamodel, etc (Giannakoglou 2002).

Acknowledgement

The first author was supported by a grant from the Secretariat of the Research Committee of the National Technical University of Athens.

For Peer Review Only

References

Alba, E. and Tomassini, M., 2002. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6 (5), 443–462.

Auger, A. and Hansen, N., 2005. A restart CMA evolution strategy with increasing population size. *In*: *The 2005 IEEE Congress on Evolutionary Computation*, Vol. 2, 1769–1776.

Coello Coello, C., Van Veldhuizen, D., and G.B., L., 2002. *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic Publishers.

Deb, K., 2001. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons.

Deb, K., *et al.*, 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6 (2), 182–197.

Désidéri, J. and Janka, A., 2003. Hierarchical parameterization for multilevel evolutionary shape optimization with application to aerodynamics. *In*: G. Bugeda, J. Désidéri, J. Periaux, M. Schoenauer and G. Winter, eds. *International Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems — EUROGEN 2003*.

Doorly, D.J. and Peiró, J., 1997. Supervised parallel genetic algorithms in aerodynamic Optimisation. *In*: *13th AIAA Computational Fluid Dynamics Conference* AIAA-1997-1852, Snowmass Village, CO, USA.

Doorly, D.J., Peiró, J., and Spooner, S., 1999. Design optimisation using distributed evolutionary methods. *In*: *37th Aerospace Sciences Meeting and Exhibit* AIAA-1999-111, Reno, NV, USA.

Drela, M. and Giles, M.B., 1987. Viscous-inviscid analysis of transonic and low Reynolds number airfoils. *AIAA Journal*, 25 (10), 1347–1355.

Duvigneau, R., Chaigne, B., and Desideri, J., Multi-level parameterization for shape optimization in aerodynamics and electromagnetics using a particle swarm optimization algorithm. , 2006. , Technical report RR-6003, INRIA.

Eby, D., *et al.*, 1998. Evaluation of injection island GA performance on flywheel design optimization. *In*: *Proceedings of the 3rd Conference on Adaptive Computing in Design and Manufacturing* Plymouth, UK: Springer-Verlag, 121–136.

Emmerich, M., Giannakoglou, K., and Naujoks, B., 2006. Single and multi-objective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4), 421–439.

Giannakoglou, K.C., 2002. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38 (1), 43–76.

Haykin, S., 1999. *Neural networks: A comprehensive foundation*. New Jersey, USA: Prentice Hall.

Herrera, F. and Lozano, M., 2000. Gradual distributed real-coded genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4 (1), 43–63.

Herrera, F., Lozano, M., and Moraga, C., 1999. Hierarchical distributed genetic algorithms. *International Journal of Intelligent Systems*, 14 (9), 1099–1121.

Jin, Y., 2005. A Comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal – A Fusion of Foundations, Methodologies and Applications*, 9 (1), 3–12.

Kampolis, I. and Giannakoglou, K., 2008. A Multilevel approach to single- and multiobjective aerodynamic optimization. *Computer Methods in Applied Mechanics and*

For Peer Review Only

Engineering, 197 (33-40), 2963–2975.

Kampolis, I., *et al.*, 2007. Multilevel optimization strategies based on metamodel–assisted evolutionary algorithms, for computationally expensive problems. *In*: *2007 Congress on Evolutionary Computation CEC07* IEEE Press, 4116–4123.

Karakasis, M. and Giannakoglou, K., 2003. Inexact Information aided, Low-Cost, Distributed Genetic Algorithms for Aerodynamic Shape Optimization. *International Journal for Numerical Methods in Fluids*, 43 (10–11), 1149–1166.

Karakasis, M. and Giannakoglou, K., 2005. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38 (8), 941–957.

Karakasis, M., Koubogiannis, D., and Giannakoglou, K., 2007. Hierarchical distributed evolutionary algorithms in shape optimization. *International Journal for Numerical Methods in Fluids*, 53, 455–469.

Knowles, J. and Corne, D., 2000. M-PAES: A memetic algorithm for multiobjective optimization. *In*: *Proceedings of the 2000 Congress on Evolutionary Computation CEC00* IEEE Press, 325–332.

Langdo, W. and Poli, R., 2005. Evolving problems to learn about particle swarm and other optimisers. *In*: *CEC-2005* IEEE Press, 81–88.

Lim, D., *et al.*, 2008. Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation, in press.*

Lin, S.C., Punch, W.F., and Goodman, E.D., 1994. Coarse-grain parallel genetic algorithms: categorization and new approach. *In*: *Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing*, 28–37.

Muyl, F., Dumas, L., and Herbert, V., 2004. Hybrid method for aerodynamic shape optimization in automotive industry. *Journal of Computers and Fluids*, 33 (5-6), 849–858.

Poloni, C., *et al.*, 2000. Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186 (2), 403–420.

Sefrioui, M. and Périaux, J., 2000. A hierarchical genetic algorithm using multiple models for optimization. *In*: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J.M. Guervós and H.P. Schwefel, eds. *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature - PPSN VI*, Vol. 1917 of *Lecture Notes in Computer Science* Paris, France: Springer, 879–888.

Srinivas, N. and Deb, K., 1995. Multiobjective optimization using nondominated sorting in genetic Algorithms. *Evolutionary Computation*, 2 (3), 221–248.

Suganthan, P., *et al.*, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. , 2005. , Technical report 2005005, Nanyang Technological University.

Zhou, Z., *et al.*, 2007. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37 (1), 66–76.

Zitzler, E., Brockhoff, D., and Thiele, L., 2007. The hypervolume INdicator revisited: On the design of Pareto-compliant indicators via weighted integration. *In*: S. Obayashi and et al., eds. *Conference on Evolutionary Multi-Criterion Optimization (EMO 2007)*, Vol. 4403 of *LNCS* Berlin: Springer, 862–876.

Zitzler, E., Laumans, M., and Thiele, L., 2002. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. *In*: *Eurogen 2001, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems* Barcelona: CIMNE, 19–26.

For Peer Review Only

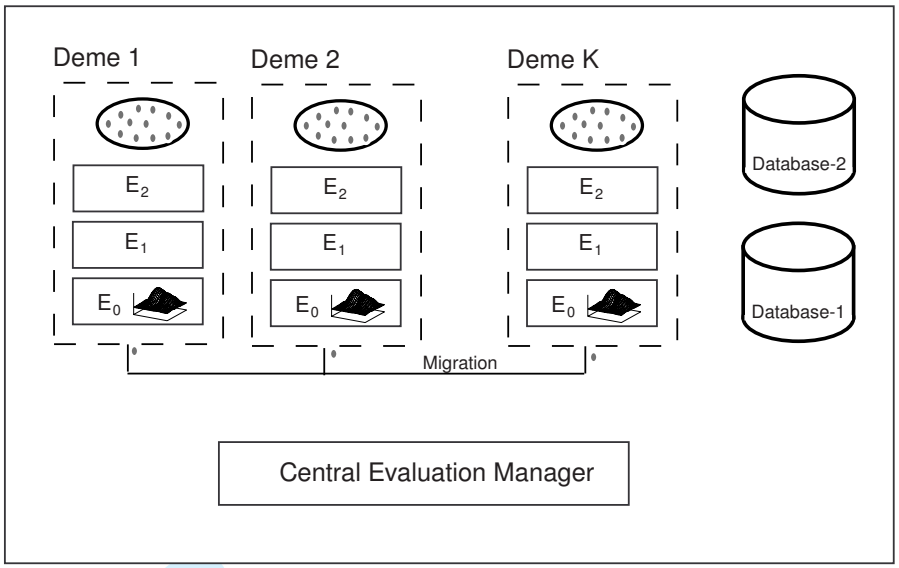


Figure 1. The proposed distributed hierarchical EA ($DHEAm$) with one metamodel (E_0) and two problem–specific tools (E_1, E_2).

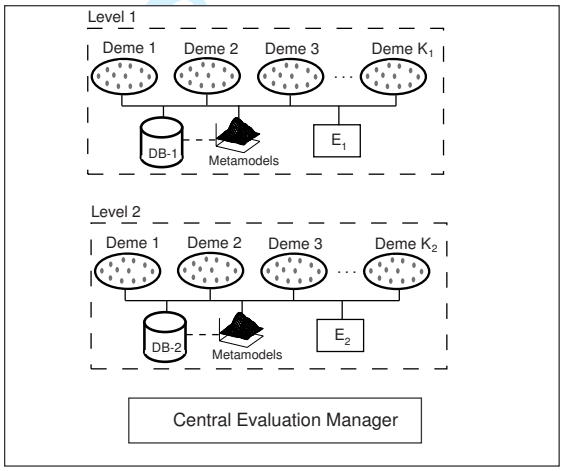


Figure 2. The hierarchical distributed metamodel–assisted EA as proposed in (Kampolis *et al.* 2007, Kampolis and Giannakoglou 2008). Each level employs a distributed EA, where the metamodels are used to pre–evaluate the population members.

For Peer Review Only

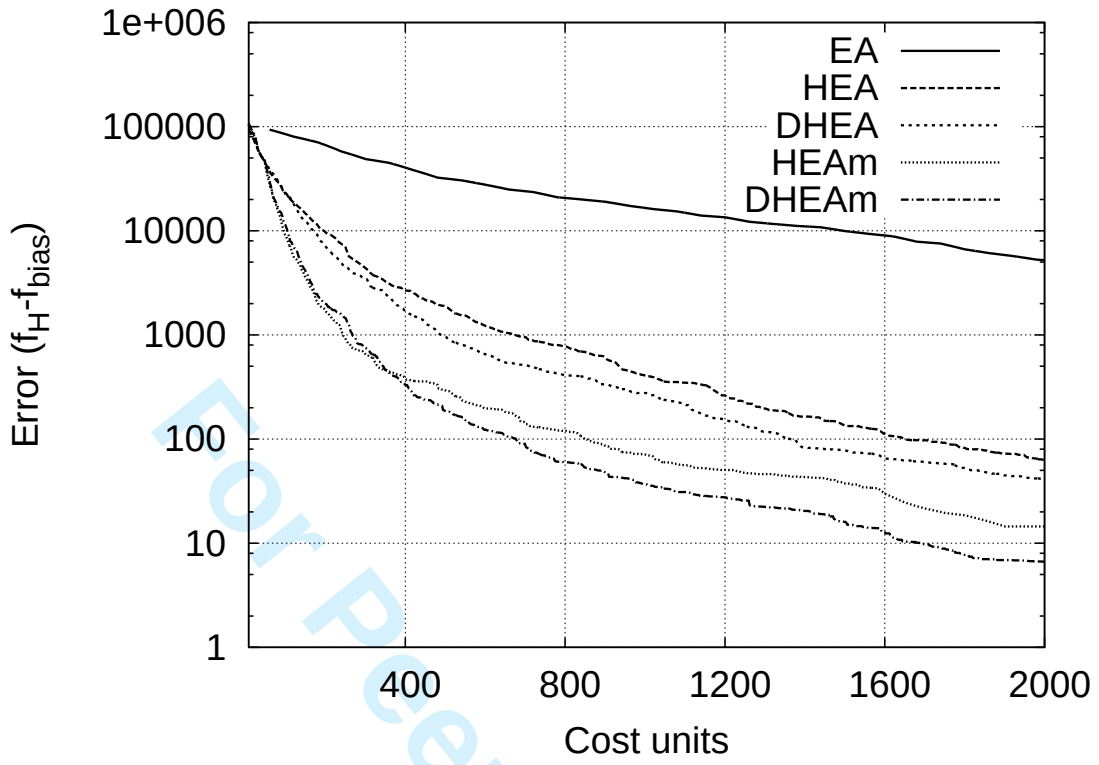


Figure 3. Sphere function minimization: Evolution of the function (without including f_{bias}) value of the best–so–far solution for 2000 CPU cost units.

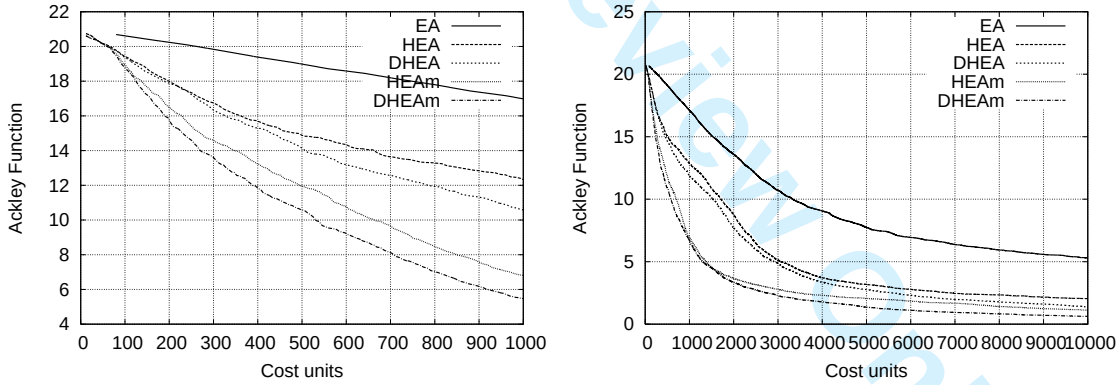


Figure 4. Ackley's function minimization: Evolution of the function value of the best–so–far solution for 1000 (left), 10000 (right) CPU cost units.

For Peer Review only

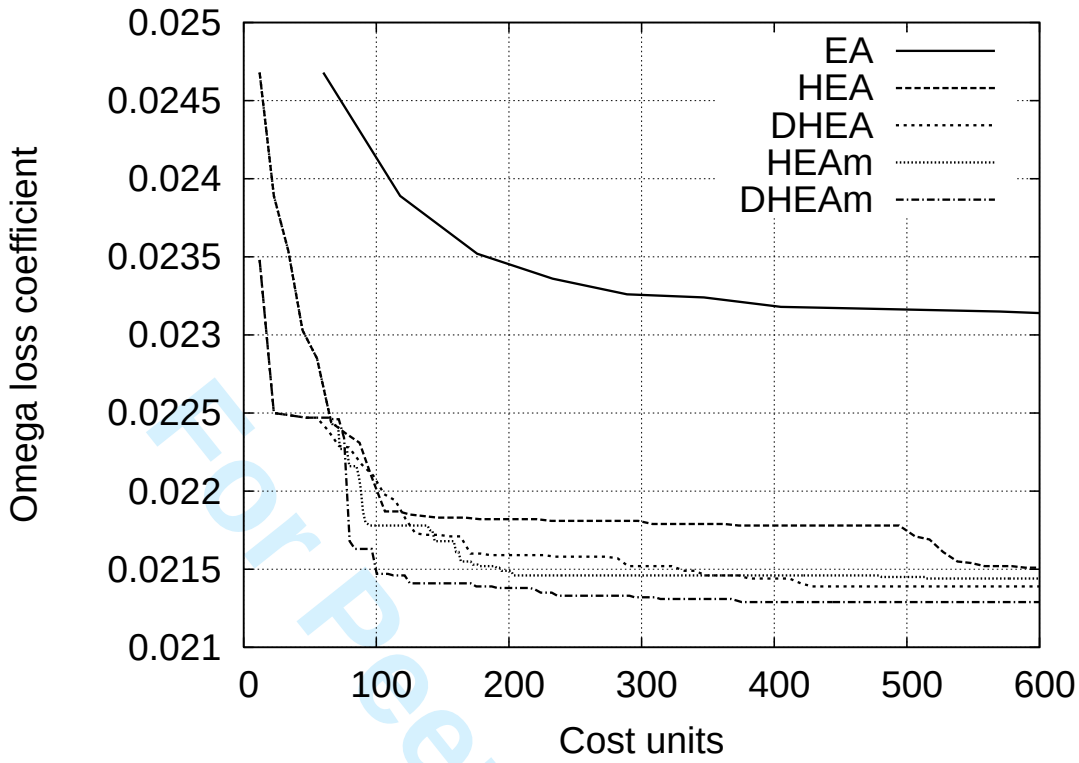


Figure 5. Single–objective compressor blade airfoil design: Evolution of the ω coefficient in terms of CPU cost units, using the five algorithms.

For Peer Review Only

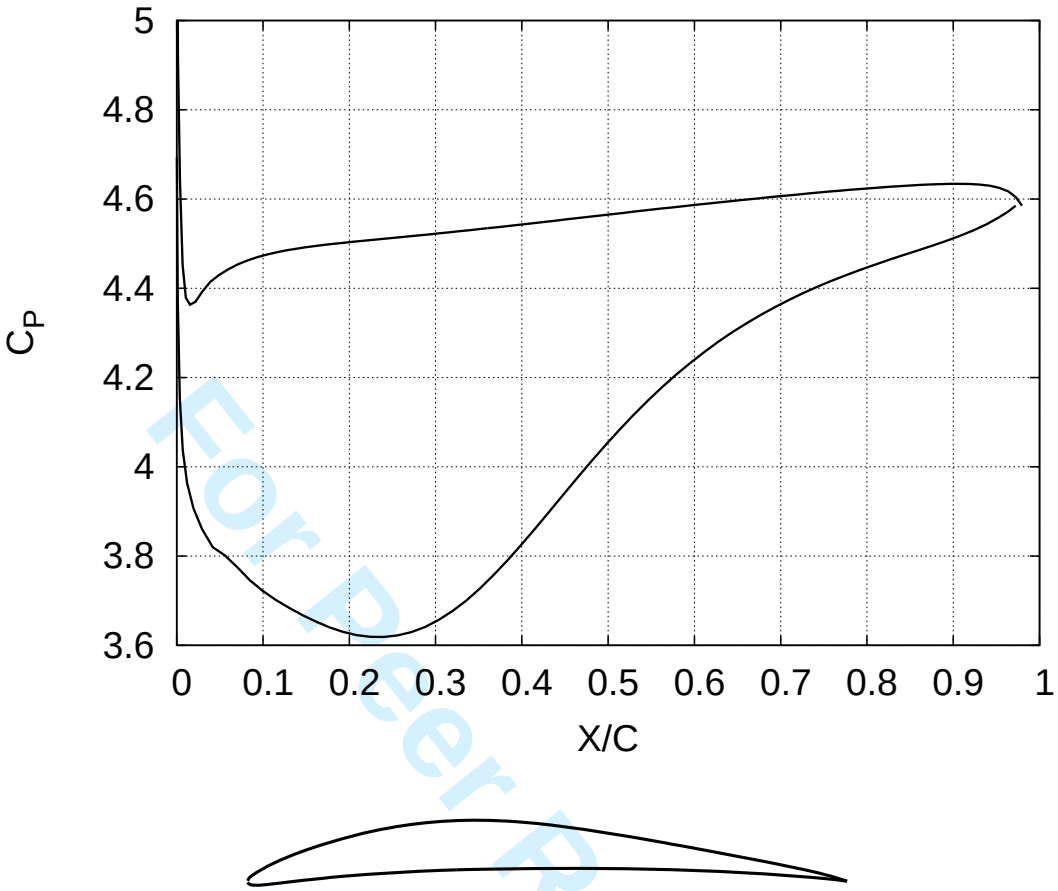


Figure 6. Single–objective compressor blade airfoil design: Pressure coefficient (c_P) distribution (top) and airfoil contour (bottom) of the optimal airfoil obtained by $DHEAm$. Plotting other airfoil shapes, resulted by the remaining four variants at the same CPU cost, is not useful since there are no visible differences.

For Peer Review Only

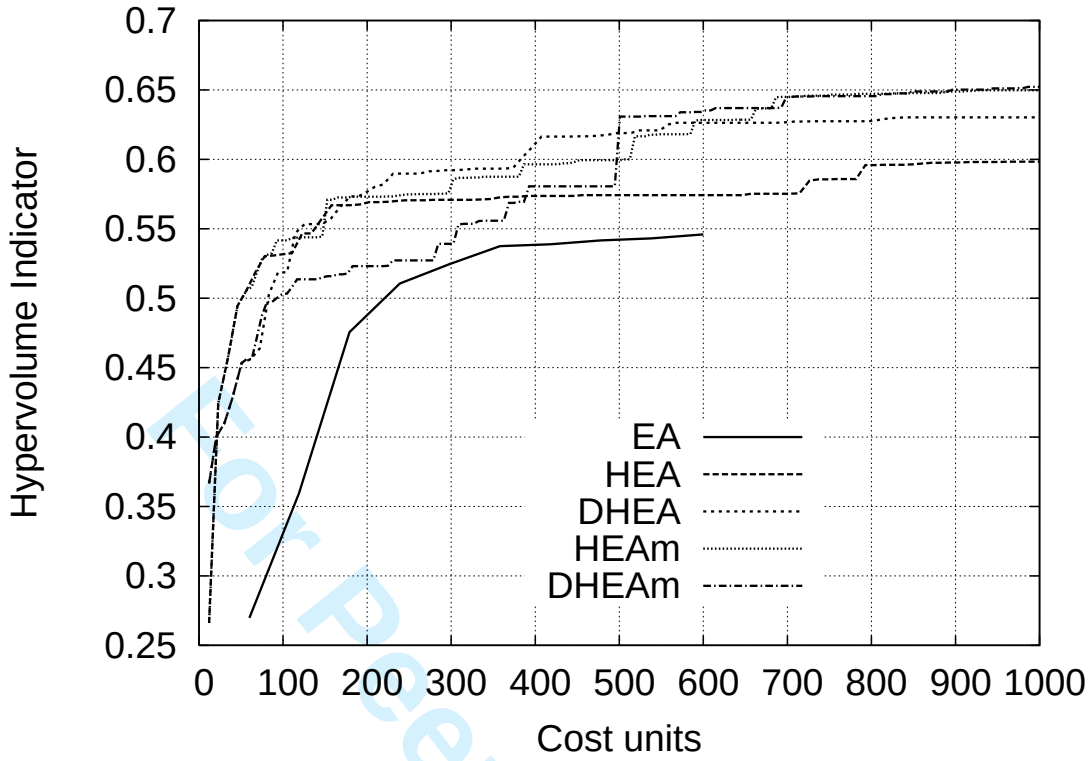


Figure 7. Two–objective compressor blade airfoil design: Evolution of the hypervolume indicator for the five algorithms used using the NSGA–2 technique.

For Peer Review Only

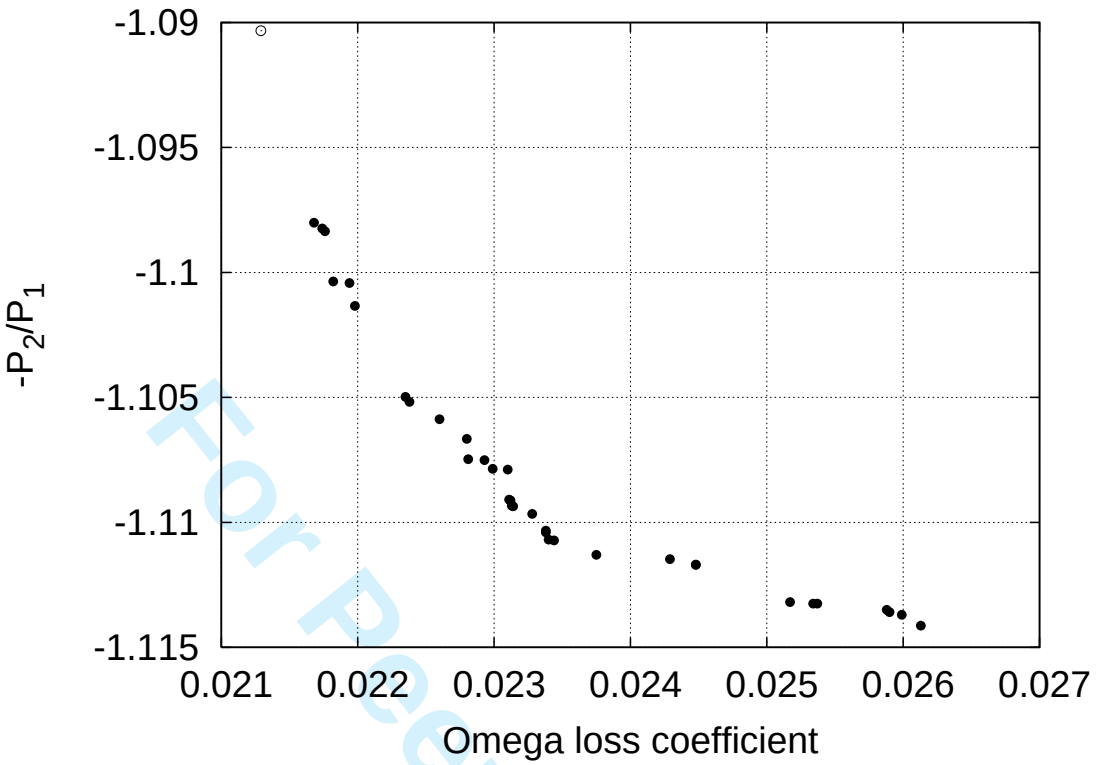


Figure 8. Two–objective compressor blade airfoil design: Front of non–dominated solutions computed using *DHEAm* employing the NSGA–2 technique (filled circles). The single–objective solution (loss coefficient minimization) is represented by the empty circle.

For Peer Review Only

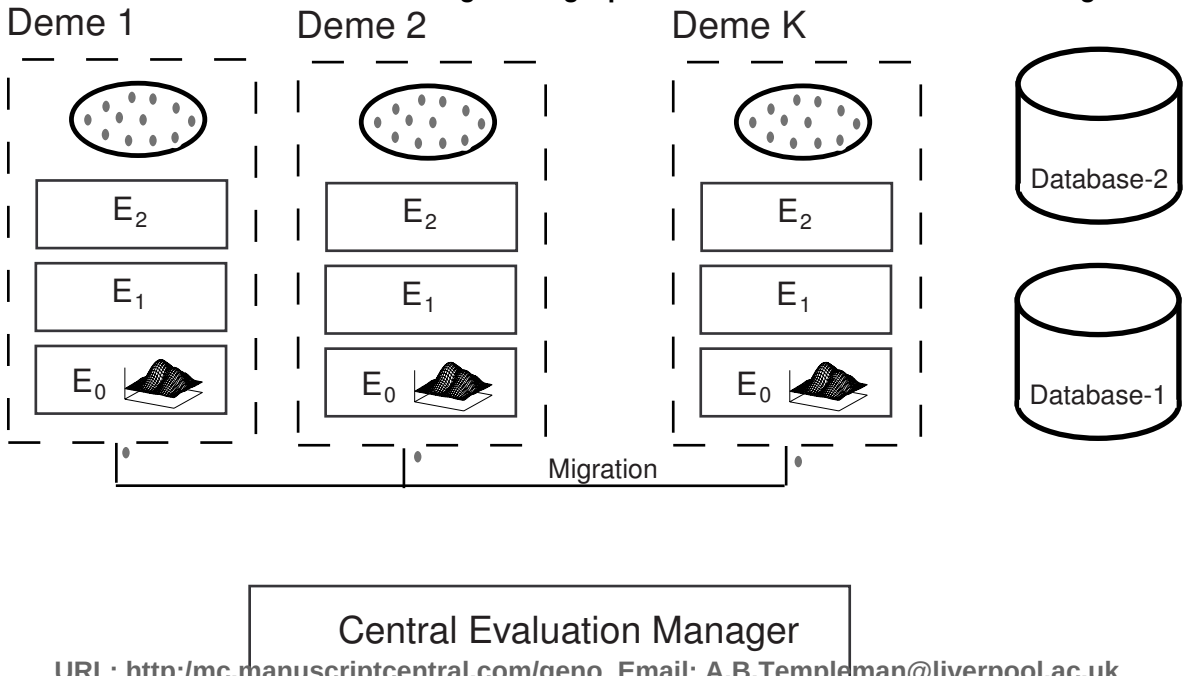
Table 1. Sphere Function. Best solution for $N = 30$ design variables and 2000 CPU cost units for the five algorithms.

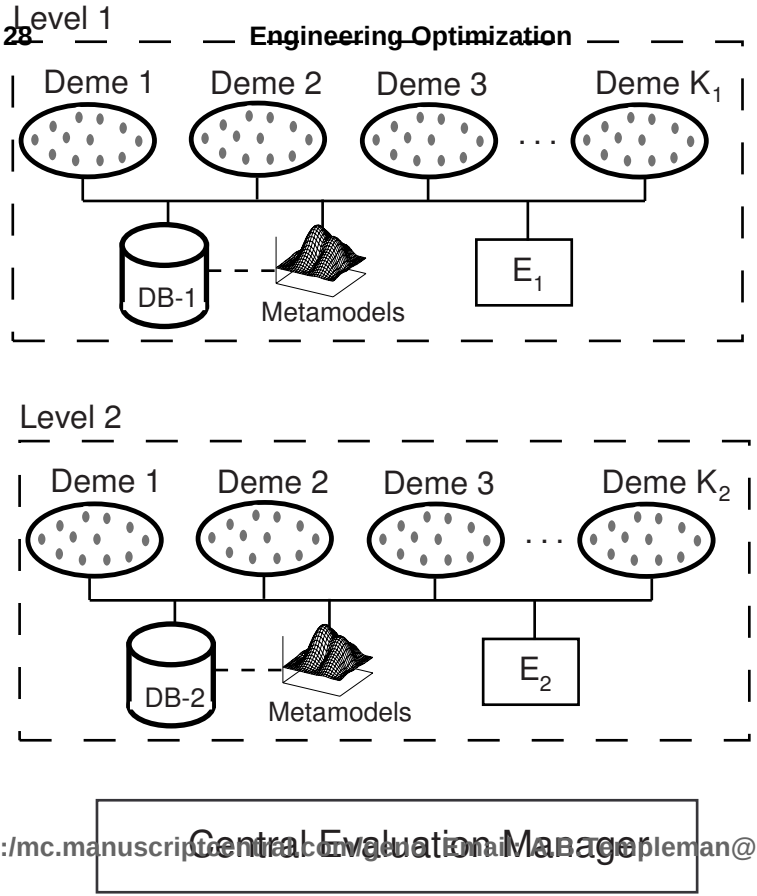
	EA	HEA	DHEA	HEAm	DHEAm
1st (Best)	2903.15	-409.82	-427.84	-442.79	-446.849
7th	3049.40	-408.74	-419.80	-440.57	-446.808
13th (Median)	4422.88	-385.45	-412.31	-439.98	-445.242
19th	6373.06	-377.25	-402.52	-428.87	-439.893
25th (Worst)	6715.98	-340.09	-389.84	-422.37	-437.368
Average	4769.30	-386.97	-409.55	-435.52	-443.342
Std Deviation	1616.67	26.38	13.46	8.0553	3.89376
t_0	-	17.467	4.176	3.2395	4.7894

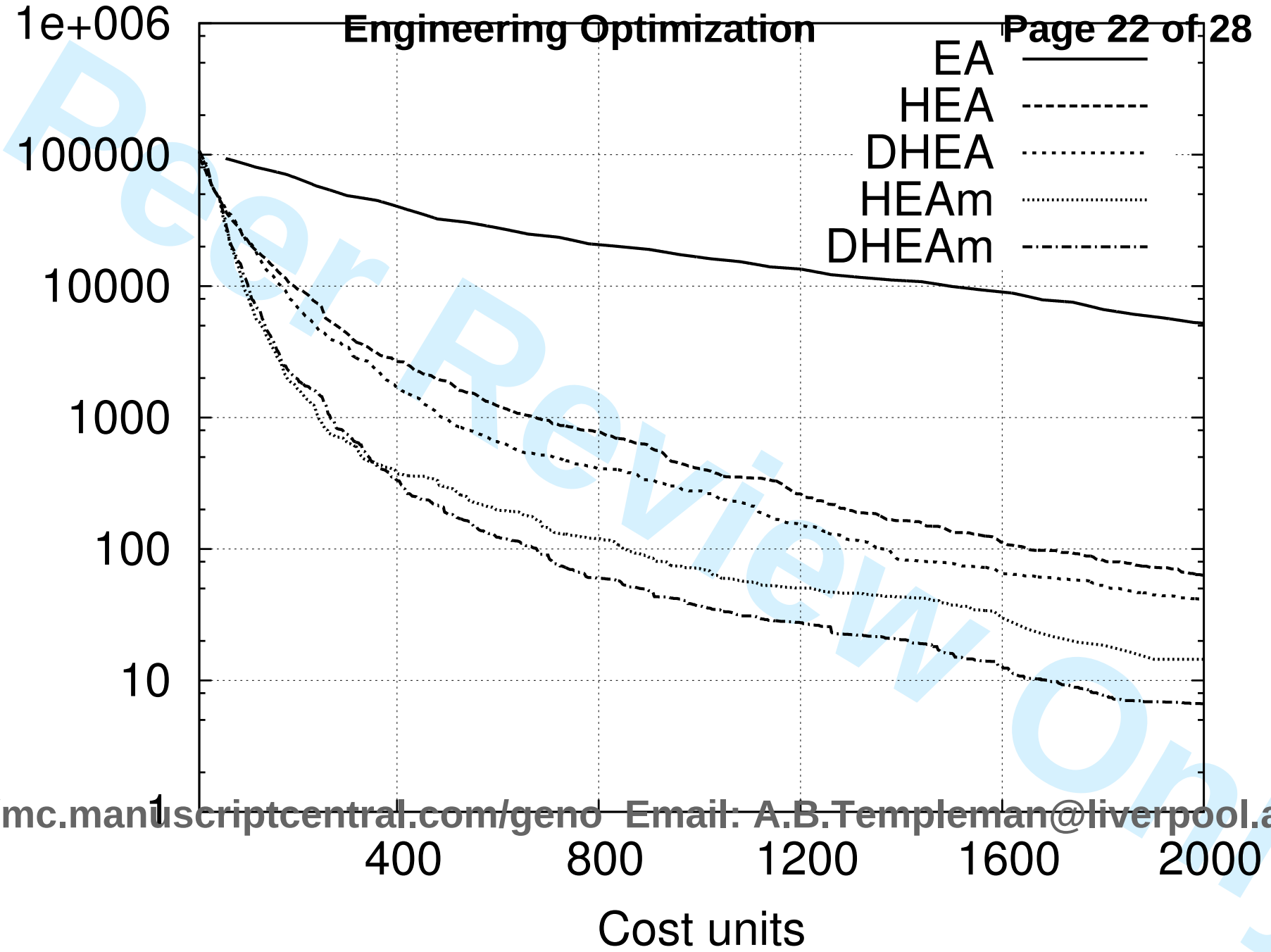
Table 2. Ackley Function. Best solution for $N = 30$ design variables and maximum CPU cost units of 1000, 5000 and 10000 units, for the five algorithms.

		EA	HEA	DHEA	HEAm	DHEAm
1000	1st (Best)	14.2361	9.0877	8.9152	4.4887	3.9754
	7th	16.1185	11.9581	10.3738	5.6511	4.8352
	13th (Median)	17.3580	12.4628	10.5892	6.8697	5.4759
	19th	17.6348	13.1168	11.1738	8.0001	5.8205
	25th (Worst)	18.2104	13.7236	11.9378	8.6807	8.5127
	Average	16.9821	12.3295	10.5878	6.8115	5.4711
	Std Deviation	0.9968	1.0419	0.8491	1.2619	0.8877
	t_0	-	17.7110	7.0976	13.5989	4.7584
5000	1st (Best)	6.5680	2.2180	1.8047	1.5016	0.9873
	7th	7.3653	2.9733	2.5453	1.9224	1.1534
	13th (Median)	7.8473	3.1292	2.7206	2.1150	1.3268
	19th	8.0671	3.4242	2.9990	2.1888	1.5734
	25th (Worst)	9.5181	3.9006	3.4317	2.5446	1.8286
	Average	7.7423	3.1797	2.7653	2.0511	1.3588
	Std Deviation	0.7136	0.4204	0.3943	0.2451	0.2630
	t_0	-	30.1733	3.9379	8.4257	10.5475
10000	1st (Best)	4.3291	1.6436	0.7870	0.6795	0.3657
	7th	5.0120	1.9051	1.2427	0.9730	0.4960
	13th (Median)	5.1978	2.0253	1.3791	1.1359	0.6082
	19th	5.4685	2.1894	1.5103	1.2697	0.6837
	25th (Worst)	6.7599	2.6027	1.9533	1.7096	1.0634
	Average	5.2831	2.0463	1.3370	1.1297	0.6260
	Std Deviation	0.5083	0.2142	0.2518	0.2438	0.1800
	t_0	-	32.1411	11.7519	3.2395	9.1037

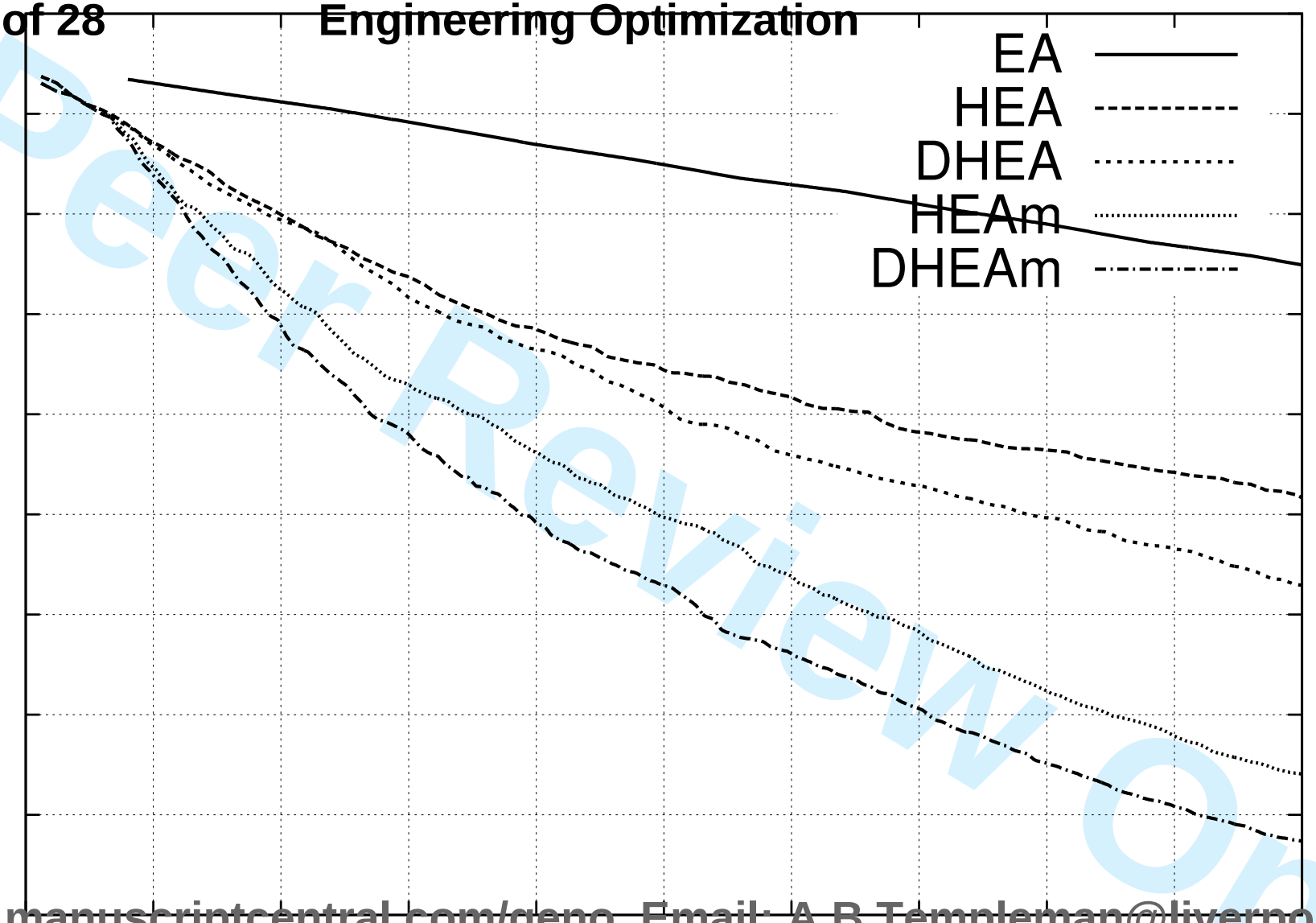
For Peer Review Only


Deme 1
Deme 2
Deme K
E_2
E_1
E_0
Database-2
Database-1
Migration
Central Evaluation Manager

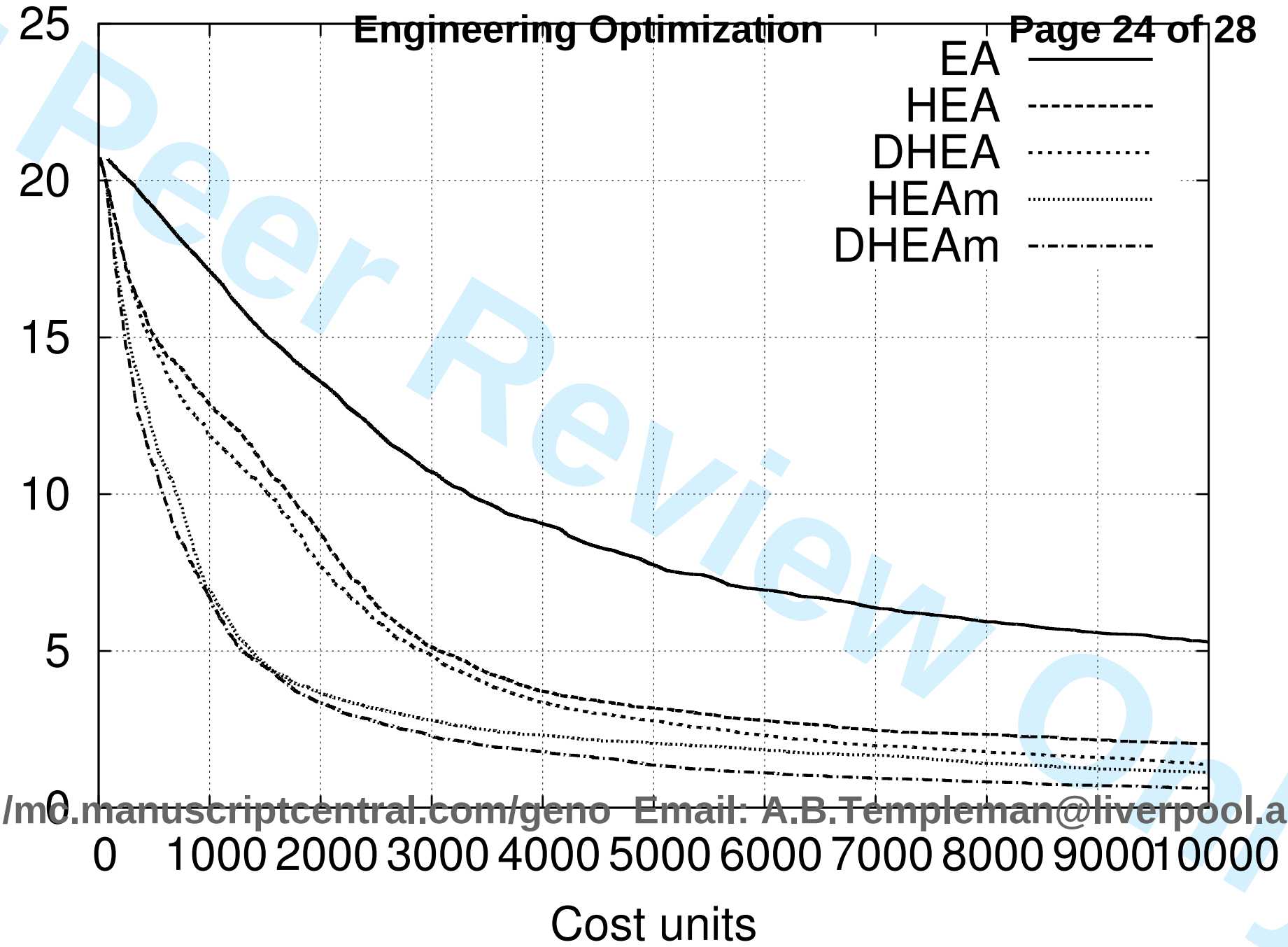

Level 1
Deme 1
Deme 2
Deme 3
Deme K_1
DB-1
Metamodels
E_1
Level 2
Deme 1
Deme 2
Deme 3
Deme K_2
DB-2
Metamodels
E_2
Central Evaluation Manager



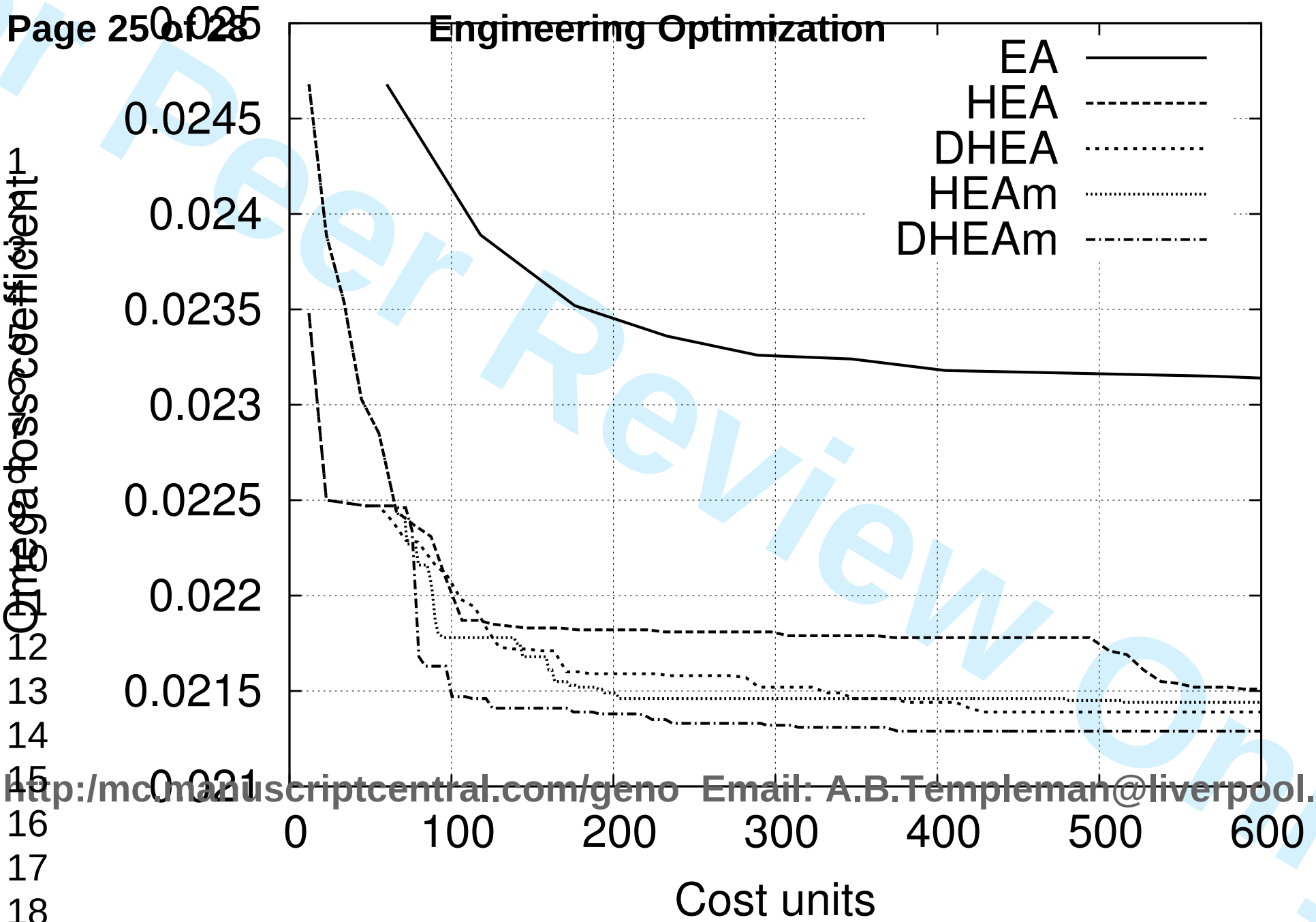
For Peer Review Only



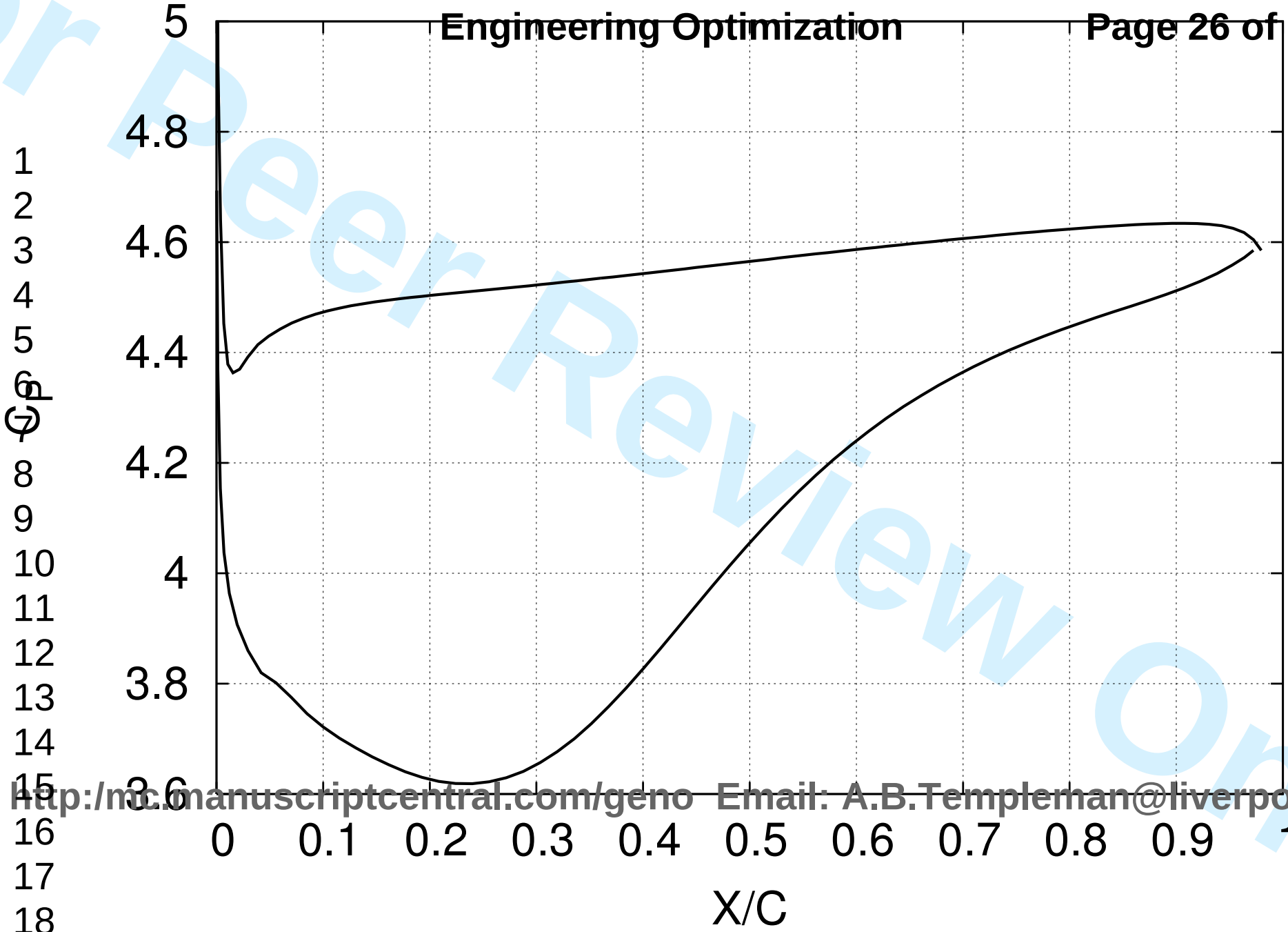
For Peer Review Only



For Peer Review Only



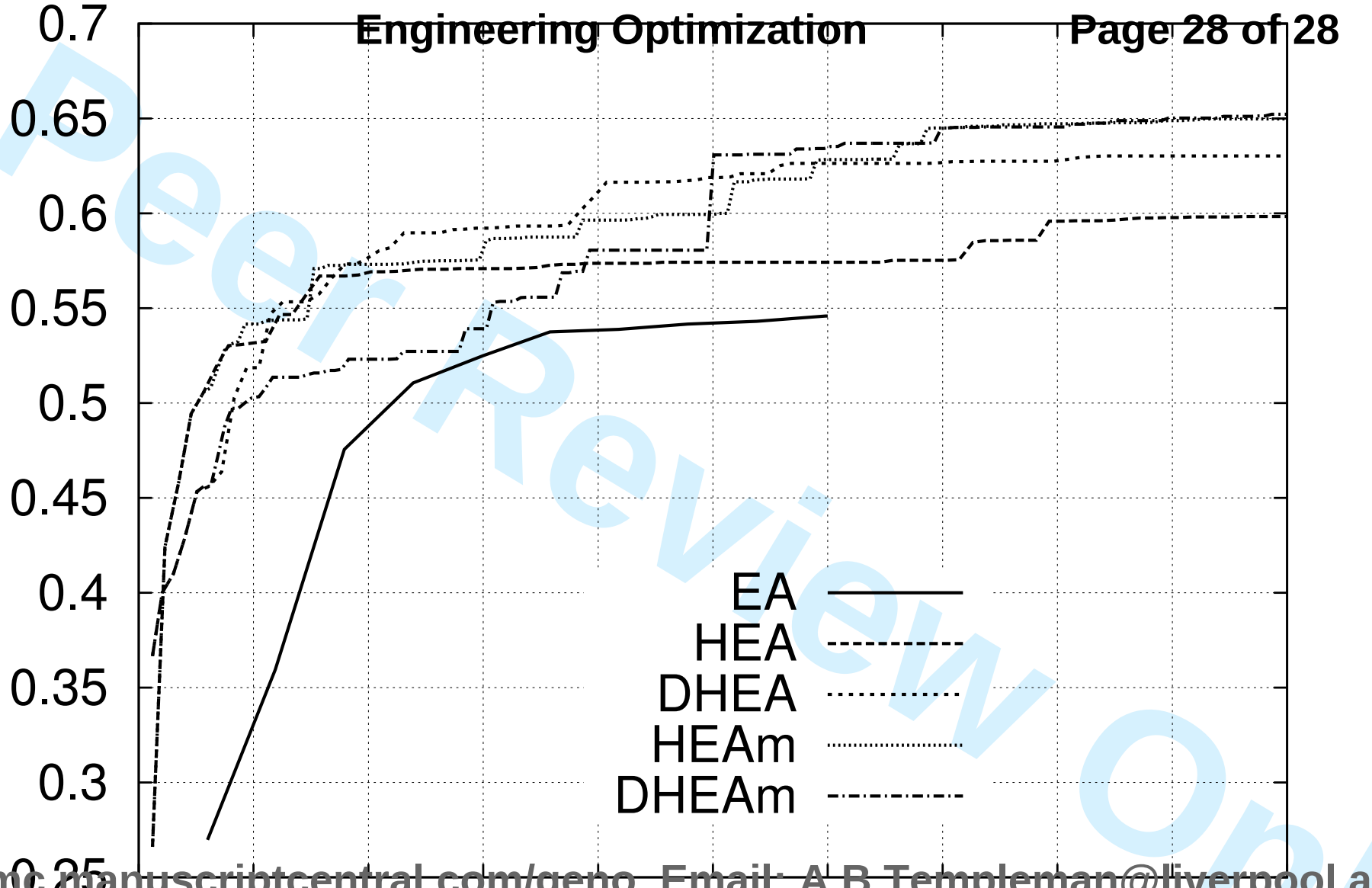
For Peer Review Only



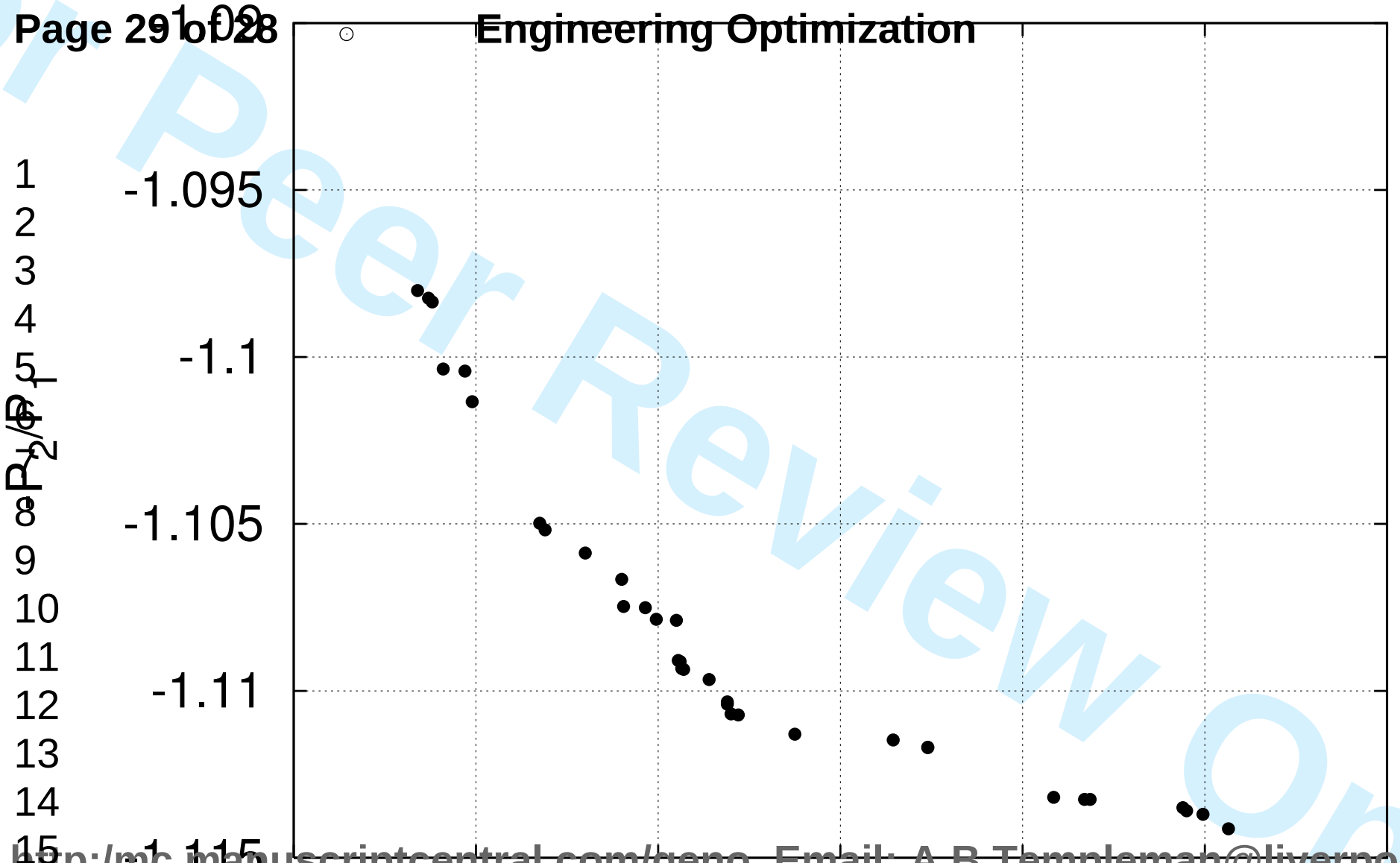
For Peer Review Only

otcentral.com/geno Email: A.B.Ten

Targeting Optimization



For Peer Review Only



For Peer Review Only