On the Anytime Behavior of IPOP-CMA-ES

Manuel López-Ibáñez, Tianjun Liao, and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{manuel.lopez-ibanez,tliao,stuetzle}@ulb.ac.be

Abstract. Anytime algorithms aim to produce a high-quality solution for any termination criterion. A recent proposal is to improve automatically the anytime behavior of single-objective optimization algorithms by incorporating the hypervolume, a well-known quality measure in multi-objective optimization, into an automatic configuration tool. In this paper, we show that the anytime behavior of IPOP-CMA-ES can be significantly improved with respect to its default parameters by applying this method. We also show that tuning IPOP-CMA-ES with respect to the final quality obtained after a large termination criterion leads to better results at that particular termination criterion, but worsens the performance of IPOP-CMA-ES when stopped earlier. The main conclusion is that IPOP-CMA-ES should be tuned with respect to the anytime behavior if the exact termination criterion is not known in advance.

Keywords: Anytime algorithms, automatic parameter tuning, continuous optimization.

1 Introduction

In many practical situations, an optimization algorithm may be terminated at any time, and, hence, it should return as high-quality solutions as possible for a wide range of possible termination criteria. Algorithms that better satisfy this property are said to have better anytime behavior [14].

When designing a new algorithm or tuning its parameters, the classical way to assess its anytime behavior is either by comparing plots of the solution-quality over time, called SQT curves [5], or by measuring performance at a different number of targets, for example, measuring solution quality after a given number of function evaluations. The benefit of the graphical comparison of SQT plots is that one gets the whole picture and it is less biased by the choice of the targets. However, a graphical comparison is intrinsically subjective. In contrast, measuring performance at different targets is an objective comparison. However, one still needs to aggregate the possibly conflicting results for each target in order to compare multiple algorithms. In this paper, we apply a new alternative, which consists in evaluating the anytime behavior as a bi-objective optimization problem. In particular, the hypervolume, a well-known quality measure in multi-objective optimization, may be used to assign a single numerical value to the anytime behavior of an algorithm's run. This technique allows us to apply automatic configuration tools for automatically improving the anytime behavior of optimization algorithms.

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 357–366, 2012.

CMA-ES [3] is a state-of-the-art algorithm for continuous optimization. Recently, a variant of CMA-ES with incremental population has been proposed [1]; we refer to this variant as IPOP-CMA-ES. The authors of IPOP-CMA-ES show that it outperforms the classical CMA-ES with restarts that keeps the population size fixed for a wide range of functions and allocated number of function evaluations. Therefore, one can say that IPOP-CMA-ES shows already a good anytime behavior. In this paper, we show that the anytime behavior of IPOP-CMA-ES can be further improved by combining automatic algorithm configuration tools and the hypervolume measure. Moreover, we also report results on tuning IPOP-CMA-ES for a specific termination criterion. The resulting configuration of IPOP-CMA-ES obtains better final quality than the default configuration and the configuration tuned for anytime behavior, but performs substantially worse if interrupted earlier than the specific termination criterion. Therefore, our results indicate that if the specific termination criterion is not known in advance or there is a high chance that IPOP-CMA-ES may be interrupted earlier, then it is better to tune IPOP-CMA-ES according to anytime behavior rather than using the default settings or tuning for a specific termination criterion.

2 Anytime Optimization

Anytime optimization algorithms may be terminated at any moment during their run, and they return a solution that is closer to the optimal the more time they were allowed to run [14]. In fact, most stochastic local search algorithms match this definition. The ideal anytime optimization algorithm would return a solution as close as possible to the optimal at any moment during its run. Hence, algorithms closer to this ideal have better anytime behavior.

One of the goals of adapting parameter settings at run-time is to adapt the exploration and exploitation trade-off to the amount of computation time allowed. Such algorithms converge very quickly to a good solution or local optimum, and then, if more time is allowed, explore more thoroughly the search space to find better solutions. Although algorithms that adapt their parameters, such as IPOP-CMA-ES, purport to remove the need to tune the parameters that are adapted, the adaptation methods introduce parameters that are subject to fine-tuning. In fact, it has been shown that automatically tuning these parameters may considerably improve the final quality obtained by IPOP-CMA-ES on diverse and difficult benchmarks [6, 7, 12]. One may argue, however, that this fine-tuning probably makes the algorithm more dependent on the particular termination criterion used in the tuning, in other words, it worsens its anytime behavior. Hence, it would be desirable to fine-tune the parameters of such algorithms in a way that is not specific to a particular termination criterion.

3 Automatically Improving Anytime Behavior

The anytime behavior of an algorithm may be modeled as a bi-objective optimization problem in terms of Pareto-optimality. In this model, the output of a run of an algorithm is a set of points in the *time* $\times$ *quality* space representing every instant that the algorithm found a solution closer to the optimal. This set of points is by definition mutually nondominated, that is, there is no point in the set that is better than another point in one criterion and not worse in the other. According to this model, we can say that a run of algorithm A has a better anytime behavior than a run of algorithm B, if the output of A is better than the output of B in the Pareto sense, that is, if all points from B are dominated by at least one point from A, and there is no point from A that is dominated by a point from B. In practice, the SQT curves of high-performing algorithms will often cross, and, hence, their outputs are often incomparable in the strict Pareto sense. This is a usual case in multi-objective optimization, and, frequently, unary quality measures are used to compare nondominated sets. Among the quality measures available, the hypervolume is the only one always able to detect whether one nondominated set is not worse than another [15]. When all objectives are minimized, the hypervolume of a nondominated set is the area of the objective space that is bounded below by the set and above by a reference point that should be the same for all sets under comparison. Thus, a larger hypervolume corresponds to a better quality.

Using the above model, López-Ibáñez and Stützle [11] have proposed to integrate the hypervolume into an automatic configuration tool in order to automatically improve the anytime behavior of optimization algorithms. We show here that this technique is able to significantly improve the anytime behavior of IPOP-CMA-ES with respect to its default settings.

4 Experimental Setup

In this paper, we try to automatically improve the anytime behavior of IPOP-CMA-ES. IPOP-CMA-ES is (μ, λ)-evolution strategy that samples a new population of solutions at each iteration from a multi-variate normal distribution. The parameters of this normal distribution are adapted during the run of the algorithm in order to focus the sampling on the most promising region of the search space. IPOP-CMA-ES obtained the best performance in the special session on real parameter optimization of the 2005 IEEE Congress on Evolutionary Computation (CEC'05), and, thus, it is a state-of-the-art algorithm for continuous optimization. In our experiments, we use the C version of IPOP-CMA-ES from Hansen's webpage `http://www.lri.fr/~hansen/cmaesintro.html`. We have modified the code to handle bound constraints by clamping the variable values outside the bounds on the nearest bound value [7].

There are a number of internal parameters of IPOP-CMA-ES that are fixed in the default implementation. These are the initial population size λ_0, the number of parent solutions selected from the population μ, and the initial step-size σ_0 among others. The population size is multiplied by a factor (*ipop*) every time the algorithm is restarted. Restarts are controlled by three additional parameters: *stopTolFunHist*, which is a lower threshold on the range of the best objective function values in recent generations; *stopTolFun*, which is a lower threshold that,

Table 1. Parameters that have been considered for tuning. Given are the default values of the parameters and the continuous range we considered for tuning. The last two columns are the parameter settings obtained for the anytime tuning (tanytime) and the tuning for the final solution quality (tfinal), respectively.

Parameter (tuning)	Internal parameter	Default	Range	Tuned tanytime	Tuned tfinal
a	Init pop size: $\lambda_0 = 4 + \lfloor a \ln(D) \rfloor$	3	$[1, 10]$	3.676	9.600
b	Parent size: $\mu = \lfloor \lambda / b \rfloor$	2	$[1, 5]$	1.750	1.452
c	Init step size: $\sigma_0 = c \cdot (B - A)$	0.5	$(0, 1)$	0.325	0.603
d	IPOP factor: $ipop = d$	2	$[1, 4]$	1.840	3.292
e	$stopTolFun = 10^e$	−12	$[-20, -6]$	−9.653	−8.854
f	$stopTolFunHist = 10^f$	−20	$[-20, -6]$	−10.000	−9.683
g	$stopTolX = 10^g$	−12	$[-20, -6]$	−9.528	−12.550

in addition to the previous range, also includes all objective function values in the last generation; and *stopTolX*, which is a lower threshold on the standard deviation of the normal distribution.

For tuning IPOP-CMA-ES, we have exposed seven parameters that directly control the internal parameters of IPOP-CMA-ES defined above. These seven parameters are given in Table 1, together with the internal parameter of IPOP-CMA-ES controlled by each of them, their default value and the range considered here for tuning. As tuner we use `irace` [9], a publicly available implementation of the automatic configuration method Iterated F-Race [2]. The budget of each run of `irace` is set to 5 000 runs of IPOP-CMA-ES. The other inputs of `irace` are the parameter ranges given in Table 1 and a set of training instances.

As benchmark instances, we consider the 19 functions from the SOCO benchmark set [4] and the 25 functions from the CEC'05 benchmark set [13]. In order to avoid over-tuning, the training set of instances used for tuning is different from the test sets used for analyzing the results of the tuning. Training instances are a subset of the functions in the SOCO benchmark, with dimension $D \in [5, 40]$. The training functions are then sampled in a random order from all possible such functions [8]. For analyzing the results, we use three test sets: 19 SOCO benchmark functions, but the 10-dimensional (SOCO-10D) and the 100-dimensional (SOCO-100D) versions, and the CEC benchmark functions with dimension 50 (CEC-50D).

We follow the protocols suggested by the authors of the SOCO and CEC benchmarks [4, 13], that is, the maximum number of function evaluations is $5\,000 \cdot D$ for the SOCO functions and $10\,000 \cdot D$ for the CEC functions. Each run of IPOP-CMA-ES is repeated 25 times on each function with different random seed. We report error values defined as $f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$, where $\boldsymbol{x}$ is a candidate solution and $\boldsymbol{x}^*$ is the optimal solution. Following the recommendation of the authors of the CEC benchmark, we use 10^{-8} as the minimum error (zero threshold), and lower values are clamped to this minimum.

5 Experimental Results

We automatically configure the parameters of IPOP-CMA-ES according to anytime behavior. For each run of IPOP-CMA-ES, every time a solution better than the best of the current run is found, we record the number of function evaluations (FEs) performed so far and the quality of the new best solution. In this manner, each run produces a nondominated set of points of quality versus FEs. We restrict the minimum number of FEs to D, that is, we start recording the solution quality after D FEs, in order to avoid the bias of the initial random sampling of IPOP-CMA-ES. All nondominated sets under comparison for the same benchmark function are normalized to the interval $[1, 2]$. Then, we compute the hypervolume of the normalized nondominated sets using $(2.1, 2.1)$ as the reference point. We integrate this procedure into irace, and use the hypervolume to evaluate each run of IPOP-CMA-ES. In this manner, we obtain a configuration of IPOP-CMA-ES called henceforth tanytime (Table 1).

Next, we run both tanytime and the default configuration of IPOP-CMA-ES (henceforth, default) on each benchmark function of the three test sets. Each run is repeated 25 times with different random seed. We compute the mean hypervolume of these runs using the same procedure described above. The results reported in Table 2 show that the tuning works, that is, the tanytime configuration obtains better (larger) hypervolume values than default in most functions, even when testing on functions with different dimensionality or from a different benchmark set. Nonetheless, it was not possible to improve the hypervolume on all functions at the same time with a single parameter setting. We performed a two-sided Wilcoxon matched-pairs signed-rank test at the 0.05 α-level, which indicates that the differences in favor of tanytime are significant in each of the three test sets. Therefore, we have found a configuration of IPOP-CMA-ES with better anytime behavior according to the hypervolume.

We assess how much this improvement is visible when evaluating the anytime behavior according to SQT curves, computed as mean error value versus FEs. Figure 1 shows the mean SQT curves, where error values are averaged over 25 runs, on a few test functions of two configurations of IPOP-CMA-ES: default and tanytime. Both axes are in logarithmic scale. Other plots are available as supplementary material [10]. The first observation is that for those functions where the SQT curve of tanytime is clearly better than the one corresponding to default, the hypervolume of tanytime is always higher, which confirms the numerical results. In the few cases where default has a higher hypervolume than tanytime, the SQT curves look like the two plots in the right column of Fig. 1.

Next, we analyze the overall quality reached at a number of termination criteria. We define termination criteria FE_1, FE_2, FE_3, FE_4, and FE_5, which correspond, respectively, to $\{1D, 10D, 100D, 1000D, 5000D\}$ FEs for SOCO functions and $\{2D, 20D,\ 200D,\ 2000D,\ 10000D\}$ FEs for CEC functions. In order to measure the overall quality, we need to summarize error values from different benchmark functions, but the range and distribution of error values varies extremely from function to function. Depending on the scenario, one could assume that the error values are comparable, and compute summary statistics directly

Table 2. Hypervolume values on the SOCO benchmark functions of dimensions 10 (10D) and 100 (100D), and on the CEC'05 benchmark functions of dimensions 50 (50D). Each number is the mean hypervolume over 25 runs.

	SOCO benchmark			
	10D		100D	
Funs	default	tanytime	default	tanytime
$f_{\mathrm{soco}1}$	1.198402	1.199056	1.185412	1.190244
$f_{\mathrm{soco}2}$	1.187217	1.186541	1.195947	1.197192
$f_{\mathrm{soco}3}$	1.209725	1.209742	1.209607	1.209758
$f_{\mathrm{soco}4}$	1.194014	1.19501	1.171771	1.18795
$f_{\mathrm{soco}5}$	1.208867	1.208925	1.204043	1.20496
$f_{\mathrm{soco}6}$	1.129921	1.125299	1.115378	1.115365
$f_{\mathrm{soco}7}$	1.209906	1.209913	1.209999	1.21
$f_{\mathrm{soco}8}$	1.206128	1.206206	1.200541	1.200248
$f_{\mathrm{soco}9}$	1.129397	1.129854	1.091057	1.113938
$f_{\mathrm{soco}10}$	1.206764	1.207165	1.208304	1.208677
$f_{\mathrm{soco}11}$	1.186655	1.201123	1.181395	1.195451
$f_{\mathrm{soco}12}$	1.209611	1.209574	1.209227	1.209338
$f_{\mathrm{soco}13}$	1.20989	1.209898	1.209687	1.209814
$f_{\mathrm{soco}14}$	1.193432	1.193597	1.15417	1.166172
$f_{\mathrm{soco}15}$	1.209983	1.209996	1.209998	1.21
$f_{\mathrm{soco}16}$	1.209613	1.209635	1.209009	1.209123
$f_{\mathrm{soco}17}$	1.209997	1.209987	1.209807	1.209899
$f_{\mathrm{soco}18}$	1.208043	1.20837	1.207676	1.20822
$f_{\mathrm{soco}19}$	1.208811	1.209075	1.209992	1.21
Num of best	4	**15**	2	**17**

	CEC (50D)	
Funs	default	tanytime
$f_{\mathrm{CEC}1}$	1.190977	1.193059
$f_{\mathrm{CEC}2}$	1.199888	1.199974
$f_{\mathrm{CEC}3}$	1.208823	1.208888
$f_{\mathrm{CEC}4}$	1.196578	1.202285
$f_{\mathrm{CEC}5}$	1.198765	1.201426
$f_{\mathrm{CEC}6}$	1.209755	1.209822
$f_{\mathrm{CEC}7}$	1.207348	1.206793
$f_{\mathrm{CEC}8}$	0.545673	0.577091
$f_{\mathrm{CEC}9}$	1.193229	1.196095
$f_{\mathrm{CEC}10}$	1.199903	1.201255
$f_{\mathrm{CEC}11}$	1.180708	1.182876
$f_{\mathrm{CEC}12}$	1.205898	1.207761
$f_{\mathrm{CEC}13}$	1.209852	1.209925
$f_{\mathrm{CEC}14}$	0.610495	0.713652
$f_{\mathrm{CEC}15}$	1.153601	1.154691
$f_{\mathrm{CEC}16}$	1.191556	1.190394
$f_{\mathrm{CEC}17}$	1.132624	1.111859
$f_{\mathrm{CEC}18}$	1.155031	1.15255
$f_{\mathrm{CEC}19}$	1.147348	1.147457
$f_{\mathrm{CEC}20}$	1.181023	1.182817
$f_{\mathrm{CEC}21}$	1.078023	1.120791
$f_{\mathrm{CEC}22}$	1.183588	1.185362
$f_{\mathrm{CEC}23}$	1.050566	1.111973
$f_{\mathrm{CEC}24}$	1.206570	1.206478
$f_{\mathrm{CEC}25}$	1.208004	1.207968
Num of best	6	**19**

on the error values, or analyze how many runs achieve a particular error value. Instead, we consider that the error values of different functions are not directly comparable, and we use a non-parametric approach based on blocking, that is, algorithm runs are ranked per function with respect to the error value, and we compute the mean rank over all test functions in each benchmark set. Fig. 2 shows the mean rank, at each termination criterion, of the default configuration and the tanytime configuration. The other configurations shown will be explained later. The plots show that tanytime configuration ranks better than the default configuration for almost all termination criteria in all benchmark sets.

Anytime Behavior vs. Final Quality. Now we consider the possibility that the default parameters of IPOP-CMA-ES may not be the best for the benchmark sets and the maximum number of function evaluations considered here. Therefore, we tune the parameters of IPOP-CMA-ES according to the final quality

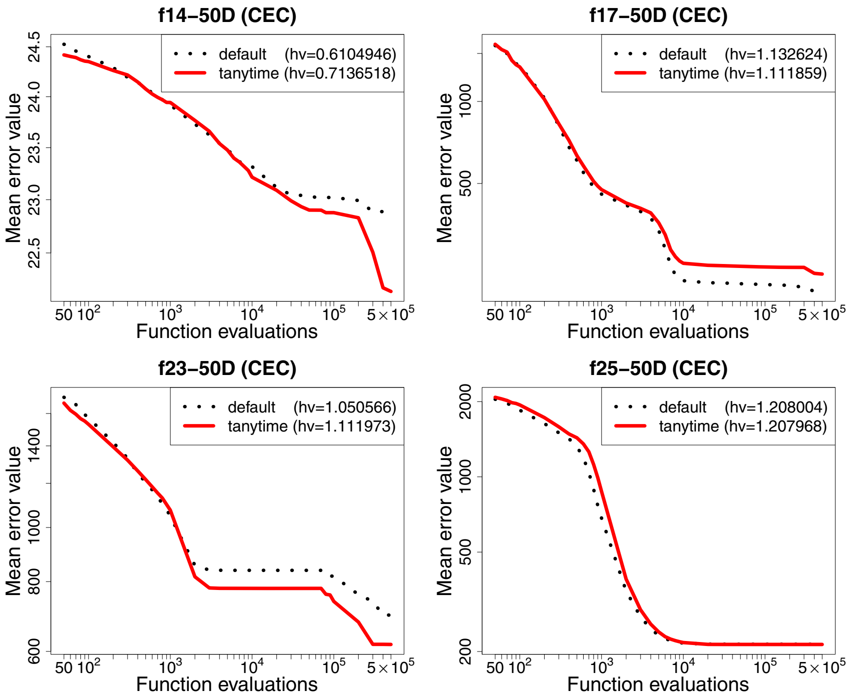

Fig. 1. SQT curves for two configurations of IPOP-CMA-ES. Plots on the left (right) show cases where tanytime obtains a better (worse) mean hypervolume than default.

obtained at the end of the run. In this way, we obtain a configuration that we call tfinal. We run this configuration on the test benchmark sets, and report the results in Fig. 2. The plots show that tfinal is able to obtain better final quality than both tanytime and default, but at the cost of worse anytime behavior. We carry out a Friedman test at each termination criterion to test for the significance of the differences between the best ranked configuration and the other two configurations. Here, we only report the results of the Friedman tests over all benchmark functions (Table 3); results per benchmark set are given as supplementary material [10]. The Friedman tests confirm these observations, that is, tfinal becomes much worse than tanytime and default if stopped earlier (termination criteria FE_1, FE_2, FE_3) than the termination criterion that was used for tuning (FE_5). The other configurations shown in the plots (and in Table 3) are explained in the next paragraph. The fact that there is a strong trade-off between final quality and anytime behavior suggests that there are still opportunities for improving the balance between fast convergence and exploration in IPOP-CMA-ES.

Hypervolume Applied to Logarithmic Transformations. The plots in Fig. 1 use a logarithmic scale in both axes, as usually done when comparing continuous optimizers. Yet, we compute the hypervolume on a linear scale, as

Table 3. Configurations of IPOP-CMA-ES ordered according to the sum of ranks obtained at each termination criterion FE_i. The numbers in parenthesis are the difference of ranks relative to the best configuration. ΔR_α is the minimum significant difference according to the Friedman test at significance level $\alpha = 0.05$. Configurations that are not significantly different from the best one are indicated in bold face.

All 63 functions (SOCO-10D, SOCO-100D, CEC-50D)		
FEs	ΔR_α	**Configurations** (ΔR)
FE1	27.34	**tany-lx-y (0),** tanytime (35.5), tany-lx-ly (38), default (116.5), tany-x-ly (158), tfinal (222)
FE2	20.39	**tany-lx-y (0),** tanytime (72.5), tany-lx-ly (78), default (98.5), tany-x-ly (214.5), tfinal (271.5)
FE3	38.32	**tanytime (0), default (26), tany-lx-ly (32), tany-lx-y (37),** tany-x-ly (57.5), tfinal (120.5)
FE4	32.28	**tfinal (0), tany-x-ly (12),** tanytime (67.5), tany-lx-ly (75.5), tany-lx-y (115.5), default (125.5)
FE5	30.51	**tfinal (0), tany-x-ly (21),** tanytime (38.5), tany-lx-ly (77.5), tany-lx-y (86), default (101)

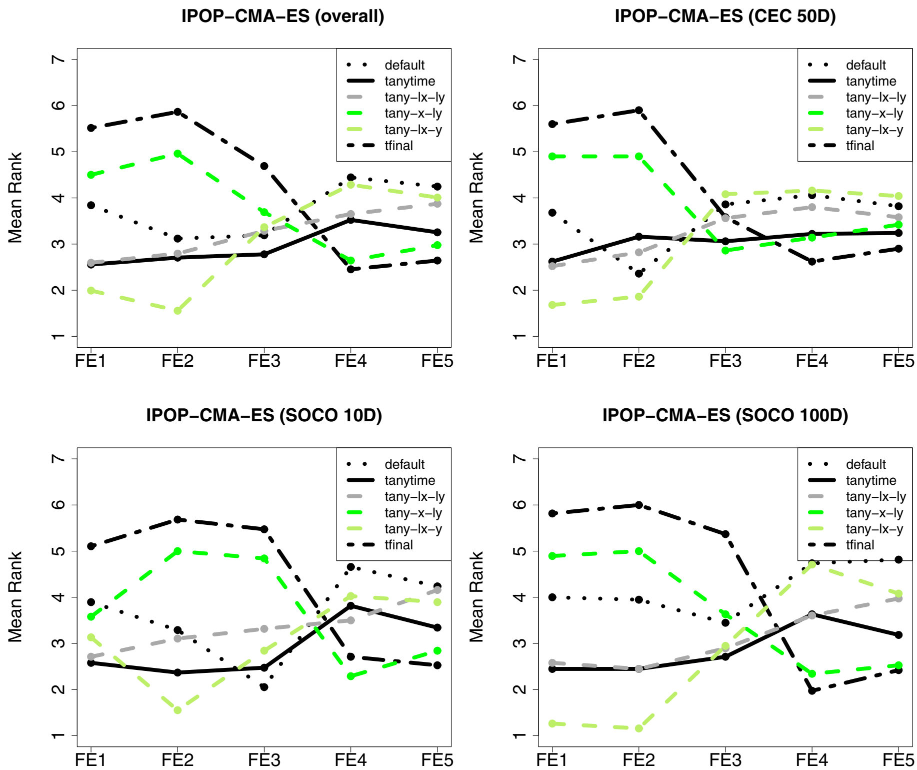

Fig. 2. Mean ranks obtained by configurations default, tanytime and tfinal at each termination criterion (FE_1 to FE_5)

commonly done in multi-objective optimization. Nonetheless, we can also apply a logarithmic scale for FEs, error values or both, before computing the hypervolume. Such transformations define a particular preference among otherwise incomparable nondominated sets, and, hence, lead to different anytime behaviors.

Fig. 2 provides an overall comparison of these alternatives (individual SQT plots are available as supplementary material [10]). Three additional configurations of IPOP-CMA-ES were obtained by tuning as described above, but using a modified hypervolume where either the number of FEs (tany-lx-y), the error values (tany-x-ly), or both (tany-lx-ly) were converted to a logarithmic scale. The plot shows that the configuration tany-lx-y (log. FE) performs better for short termination criteria, whereas the tany-x-ly (log. error values) obtains better results when running for longer FEs. Interestingly, there is no much difference between applying a logarithmic transformation to both objectives or to none of them. Our conclusion is that logarithmic transformations of only one objective (either quality or computational effort) introduce a strong bias, which should be taken into account to not defeat the purpose of tuning for anytime behavior.

6 Conclusions

In this paper, we have investigated whether the anytime behavior of IPOP-CMA-ES can be improved by automatically tuning its parameters. We have applied a recently proposed technique that integrates the hypervolume quality measure into an automatic configuration method (`irace`). Our results have shown that the anytime behavior of the default parameters of IPOP-CMA-ES can be substantially improved. Moreover, we have also shown that simply tuning IPOP-CMA-ES according to the quality achieved at a large termination criterion does improve the results at that particular termination criterion; however, it compromises the results for shorter termination criteria, becoming even worse than the default configuration of IPOP-CMA-ES. Therefore, if the specific termination criterion is not known in advance, it is better to tune IPOP-CMA-ES according to anytime behavior than for a very large termination criterion.

Our results also suggest that, despite the adaptation of the population size and the restart step in IPOP-CMA-ES, its results are not ideal in terms of anytime behavior. Therefore, we plan to investigate in the future whether the anytime behavior of IPOP-CMA-ES can be further improved by adapting other parameters or making some parameters time-varying.

Acknowledgments. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Manuel López-Ibáñez and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are a postdoctoral researcher and a research associate, respectively. Tianjun Liao acknowledges a fellowship from the China Scholarship Council. The authors also acknowledge support from the FRFC project "*Méthodes de recherche hybrides pour la résolution de problèmes complexes*". The authors have contributed equally to this work.

References

1. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proc. of CEC 2005, pp. 1769–1776. IEEE Press, Piscataway (2005)
2. Balaprakash, P., Birattari, M., Stützle, T.: Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HM 2007. LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg (2007)
3. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
4. Herrera, F., Lozano, M., Molina, D.: Test suite for the special issue of Soft Computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems (2010), `http://sci2s.ugr.es/eamhco/`
5. Hoos, H.H., Stützle, T.: Stochastic Local Search—Foundations and Applications. Morgan Kaufmann Publishers, San Francisco (2005)
6. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: An experimental investigation of model-based parameter optimisation: SPO and beyond. In: Rothlauf, F. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009, pp. 271–278. ACM Press, New York (2009)
7. Liao, T., Montes de Oca, M.A., Stützle, T.: Computational results for an automatically tuned IPOP-CMA-ES on the CEC'05 benchmark set. Tech. Rep. TR/IRIDIA/2011-022, IRIDIA, Université Libre de Bruxelles, Belgium (2011)
8. Liao, T., Montes de Oca, M.A., Stützle, T.: Tuning parameters across mixed dimensional instances: A performance scalability study of Sep-G-CMA-ES. In: Krasnogor, N., Lanzi, P.L. (eds.) GECCO (Companion), pp. 703–706. ACM Press, New York (2011)
9. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)
10. López-Ibáñez, M., Liao, T., Stützle, T.: On the anytime behavior of IPOP-CMA-ES: Supplementary material (2012), `http://iridia.ulb.ac.be/supp/IridiaSupp2012-010/`
11. López-Ibáñez, M., Stützle, T.: Automatically improving the anytime behaviour of optimisation algorithms. Tech. Rep. TR/IRIDIA/2012-012, IRIDIA, Université Libre de Bruxelles, Belgium (2012)
12. Smit, S.K., Eiben, A.E.: Beating the 'world champion' evolutionary algorithm via REVAC tuning. In: Ishibuchi, H., et al. (eds.) Proc. of CEC 2010, pp. 1–8. IEEE Press, Piscataway (2010)
13. Suganthan, P.N., Hansen, N., Liang, J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University, Singapore (2005)
14. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996)
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)