An Adaptive Population-based Candidate Search Algorithm with Surrogates for Global Multi Objective Optimization of Expensive Functions

Christine A. Shoemaker[1, 2, 3 a)] and Taimoor Akhtar[2, b)]

[1]*Department of Industrial Systems Engineering and Management, National University of Singapore.*
[2]*NUS Environmental Research Institute (NERI), National University of Singapore.*
[3]*Department of Civil and Environmental Engineering, National University of Singapore.*

a) Corresponding author: shoemaker@nus.edu.sg
b) erita@nus.edu.sg

Abstract. We propose a new algorithm, MOPLS, for efficient global Multi Objective (MO) optimization of expensive functions. MOPLS is an iterative population-based parallel surrogate algorithm that incorporates simultaneous local candidate search on surrogate models to select numerous evaluation points in each iteration. The novel iterative framework of MOPLS simultaneously selects new points for evaluation by using i) Local Radial Basis Function (RBF) approximation, ii) surrogate-assisted neighborhood candidate search, and iii) a Tabu mechanism for adaptively avoiding neighborhoods that do not improve the non-dominated solution set. MOPLS is more efficient than ParEGO, Borg, NSGA-II and NSGA-III with application to 11 test problems and a watershed calibration problem, on a budget of 600 functions evaluations.

BACKGROUND AND MOTIVATION

Evolutionary algorithms (e.g. NSGA-II [1]) and other meta-heuristics are very popular for solving Multi Objective (MO) global optimization problems. However, these algorithms may require many objective function evaluations to obtain good approximations of the Pareto front, and hence, may not be feasible for Multi-objective global optimization of computationally expensive functions.

Prior research shows that iterative surrogate optimization algorithms with Kriging [2], Radial Basis Functions [3-6] and other response surfaces as surrogates, are efficient and effective for computationally expensive problems. However, most surrogate MO algorithms proposed in prior literature perform one or a few evaluations per iteration. Efficiency of such algorithms may be improved further via use of a population/batch-based structure, such that numerous new points can be proposed and evaluated simultaneously. Past work with evolutionary algorithms [1, 7, 8] shows that evolving populations is effective in obtaining MO trade-offs that are diverse and convergent.

This study aims to combine the population-based structure of evolutionary algorithms with the surrogate and candidate search method proposed by Regis and Shoemaker [4], and present a new algorithm that proposes multiple function evaluations in each algorithm iteration, that can be evaluated simultaneously in a synchronous parallel framework for improvement in efficiency of MO global surrogate optimization.

THE MOPLS ALGORITHM

MOPLS is a **M**ulti **O**bjective **P**opulation-based **L**ocal **S**urrogate-assisted iterative candidate search algorithm designed to propose a synchronous batch of expensive function evaluations in each algorithm iteration. MOPLS follows the sequential / iterative optimization framework where surrogates are updated after each algorithm iteration. The algorithm initializes via Latin Hypercube Sampling. The sequential / iterative framework follows, that terminates after a fixed number of expensive evaluations. The following sub-sections discuss the iterative framework of MOPLS.

Overview of Iterative Algorithm Framework

The iterative framework of MOPLS has three core key components, namely, i) updating of the MOPLS "memory archive", ii) selection of a population of **P** 'centers' from the memory archive, and iii) proposal of **P** new points for function evaluation via the local surrogate-assisted candidate search (around each 'center' point of population).

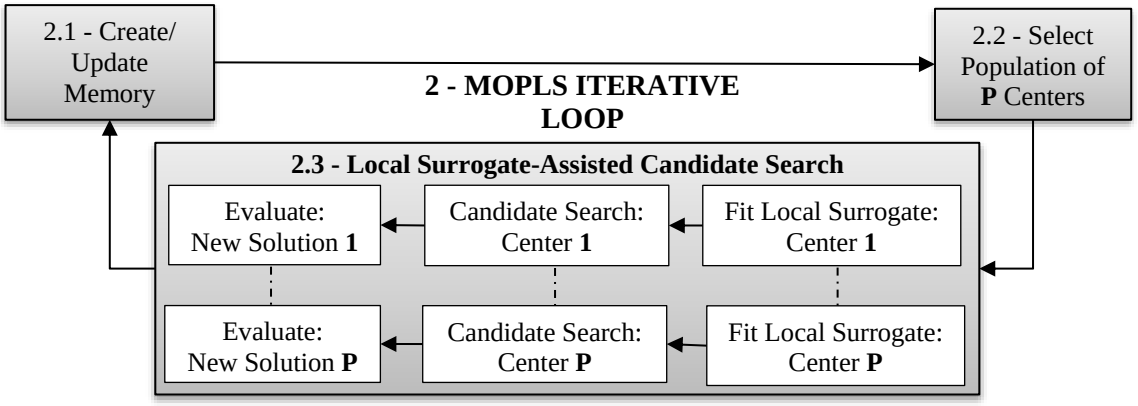

FIGURE 1. Layout of the iterative framework of MOPLS

The MOPLS Memory Archive

The MOPLS memory archive includes all previously evaluated points and sorts them according to i) their non-domination rank [1] and ii) hypervolume contribution [9]. We use this two-layered ranking system to select points as centers in the center population (see Fig. 1). However, centers selection is not entirely based on the two-layered ranking. MOPLS adaptively learns from prior algorithm progress and the memory archive maintains three additional attributes, i.e., i) failure count, ii) candidate search radius and ($\mathbf{r}_i$) iii) tabu count. If candidate search around a center (see Step 2.3 in Fig. 3) does not produce a new solution that is non-dominated in previously evaluated solutions, the candidate search has failed to improve the non-dominated set. Hence, we increase the failure count of the center by one and decrease its candidate search radius by half. After N_{fail} failures, we make the center tabu and maintain the tabu count. Making a center tabu means that it cannot be selected as center for N_{tabu} subsequent algorithm iterations (also called Tabu tenure). Hence, after N_{tabu} iterations we reset the tabu count and the point may be selected as center. N_{fail} and N_{tabu} are important algorithm parameters that control the balance between global and local search [6].

Selection of Center Points

The primary purpose of the memory archive is to assist MOPLS in adaptively learning from previous candidate searches for selection of center points in subsequent iterations, i.e., if prior candidate search around a center is not resulting in improving the non-dominated front, the algorithm stops conducting candidate search around point. Consequently, selection of population of center points (Step 2.2) is based on i) non-domination rank, ii) hypervolume contribution and iii) tabu status. The sorted solutions as per the two layered ranking of i) and ii) above, that are not tabu, are selected as centers in each iteration of MOPLS. Furthermore, an evaluated point is only added to selected centers, if previously selected centers are not in the candidate search neighborhood radius ($\mathbf{r}_i$) of the evaluated point.

Local Surrogate-Assisted Candidate Search

We perform a local surrogate-assisted candidate search around each selected 'center' to propose one new expensive point for evaluation per center (See Step 2.3 in Fig. 1). For each center ***i***, we first fit a Radial Basis Function (RBF) surrogate model with a maximum of M closest points to center ***i***. Many random candidate points are then generated via Gaussian perturbation of center ***i***. This idea of generating candidate points has been successfully introduced [4] and used in single objective surrogate global optimization algorithms [5,6]. We subsequently select one candidate point for expensive evaluation based on its approximate hypervolume improvement [3] as per the RBF surrogates.

RESULTS AND DISCUSSION

Experimental Setup

Test Problems: We test MOPLS on 11 bi-objective test problems and a watershed problem. The test problems include five ZDT problems (No. 1-4 & 6) proposed by Zitzler et al. [10] and six problems proposed by Li et al. [11]. Number of decision variables for the test problems can vary and are set to 8 and 24 for all problems to analyse the effect of changing dimensions. The watershed calibration problem is an unconstrained problem for the small Townbrook catchment in New York [12] and has 15 parameters and 2 objectives (representing different forms of model error). Simulation runtime is in order of seconds (and can be in order of minutes / hours for larger catchments).

Baseline Algorithms: We compare MOPLS against the Gaussian Process based surrogate algorithm ParEGO [2], the steady-state evolutionary algorithm Borg [8], and the popular genetic algorithms, NSGA-II [1] and NSGA-III [7].

Comparison Methodology: The focus of this study is on MO optimization of expensive problems (i.e., simulation time is in order of minutes to hours). Hence, we limit the evaluation budget of our experiments to a few hundred evaluations (400 for the test problems and 600 for the watershed problem). Since all algorithms are stochastic, we run 10 trials for each algorithm on each problem, and use the uncovered hypervolume ($\boldsymbol{H_u}$) as a measure for comparing algorithms. $\boldsymbol{H_u}$ is the difference (normalized) between the total feasible hypervolume (covered by the Pareto front) and the hypervolume covered by the non-dominated solutions of an algorithm. Low values of $\boldsymbol{H_u}$ are better.

Results

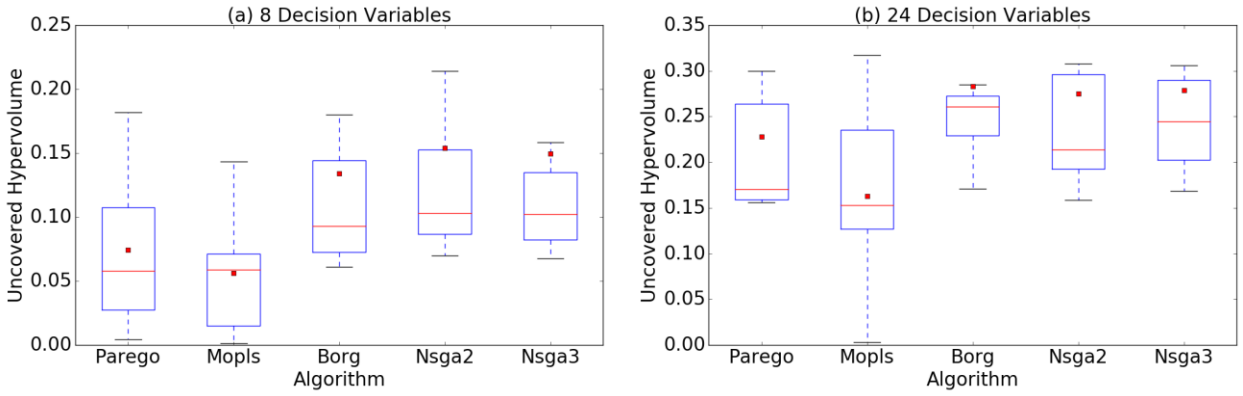

FIGURE 2. *Box plots* of average (over multiple trials) uncovered hypervolume ($\boldsymbol{H_u}$) values after 400 function evaluations for each algorithm over the eleven test problems: a) Test problems with 8 decision variables and b) Test problems with 24 decision variables. Low uncovered values are best.

Figure 2 shows the box plots of average (over multiple trials) $\boldsymbol{H_u}$ values obtained via application of each algorithm to the eleven test problems with 8 (Fig. 2a) and 24 (Fig. 2b) decision variables, after 400 evaluations. The red-square markers within each boxplot report the means (over multiple test problems) of average $\boldsymbol{H_u}$ values for each algorithm.

Figure 2 clearly indicates that MOPLS has the best overall performance (depicted by lower $\boldsymbol{H_u}$ values) amongst all algorithms for both the 8 and 24 decision variants of the test problems, within 400 evaluations. Performance of ParEGO is not far from MOPLS for the 8-variable problems (see Fig. 2a). However, as per the Wilcoxon rank-sum test (performed on $\boldsymbol{H_u}$ values for each problem after 400 function evaluations), MOPLS is 'significantly' (at 5% significance) better than ParEGO on 4 out of 11 eight-variable problems and 7 out of 11 twenty-four-variable problems. None of the other algorithms is significantly better than MOPLS on any problem. Furthermore, MOPLS is designed to be batch-parallel, since **P** points are proposed for expensive evaluation in each iteration, whereas ParEGO is serial. Hence, relative efficiency of MOPLS will be even better in parallel. For all experiments in this analysis **P=8** for MOPLS. Borg is serial, whereas NSGA2 and NSGA3 are also batch-parallel with a population size of 16.

Figure 3 summarizes the performance of algorithms on the watershed calibration problem within 600 function evaluations. Fig. 3a plots the median fronts for each algorithm and we can see that the front obtained from MOPLS dominates the fronts obtained from all other algorithms. The progress curves of Fig. 3b plot $\boldsymbol{H_u}$ values against number of function evaluations and we can see that MOPLS has better efficiency and faster convergence than all other

algorithms. Furthermore, we also see that MOPLS gets the same H_u value that ParEGO gets after 600 evaluations, within 200 evaluations. This means that MOPLS is at least 3 times faster than ParEGO on the watershed problem.

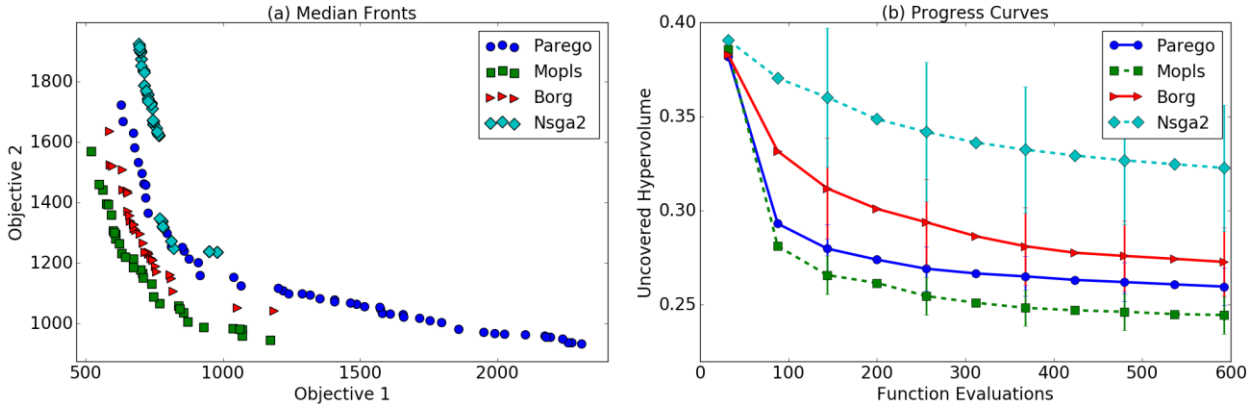

FIGURE 3. Comparison of watershed problem results: a) Visualisation of median (as per uncovered hypervolume) non-dominated fronts from all algorithms after 600 function evaluations (fronts closer to origin are better), b) Uncovered hypervolume progress for all algorithms, i.e. plots of uncovered hypervolume vs functions evaluations (low curves are better).

CONCLUSION

MOPLS is a new surrogate algorithm that is efficient and effective for MO global optimization of expensive problems. The population-based/generational iterative structure of MOPLS allows the algorithm to run in batch-parallel mode for further improving its efficiency. Comparison of MOPLS to ParEGO, and evolutionary algorithms shows that MOPLS is better than other algorithms in serial. In future, we plan to test performance of MOPLS in parallel, to test its framework with a Gaussian Process-based Bayesian surrogate and the Expected Hypervolume Improvement (EHVI) [13] criterion, and to develop a steady-state version of MOPLS for asynchronous parallelization.

ACKNOWLEDGMENTS

This work has been funded by Phase 1 of Energy and Environmental Sustainability for Megacities (E2S2) CREATE research program and by Prof. Shoemaker's startup grant from National University of Singapore (NUS). MOPLS was implemented in the PySOT [14] toolbox and we acknowledge the assistance of co-authors of PySOT.

REFERENCES

1. K. Deb, A. Pratap, S. Agarwal and T. Meyairvan, IEEE Trans. On Evol. Comp. **6**, 182-197 (2002).
2. J. Knowles, IEEE Trans. On Evol. Comp. **10**, 50–66 (2006).
3. T. Akhtar and C. A. Shoemaker, J. Glob. Optim. **64**, 17–32 (2016).
4. R. G. Regis and C. A. Shoemaker, INFORMS J on Comp. **19**, 497–509 (2007).
5. R. G. Regis and C. A. Shoemaker, Engineering Optimization **45**, 529–555 (2013).
6. T. Krityakierne, T. Akhtar and C. A. Shoemaker, J. Glob. Optim. **66**, 417–437 (2016).
7. K. Deb, and H. Jain, IEEE Trans. On Evol. Comp. **18**, 577-601 (2014).
8. D. Hadka and P. Reed, Evolutionary Computation **21**, 231-259 (2013).
9. M. Emmerich, N. Beume and B. Naujoks, *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization*, Mexico 2005, Springer (Berlin 2005), Vol. 3410, pp. 62-76.
10. E. Zitzler, K. Deb, and L. Thiele, Evolutionary Computation **8**, 173–195 (2000).
11. H. Li and Q. Zhang, IEEE Trans. On Evol. Comp. **13**, 284–302 (2009).
12. B. A. Tolson and C. A. Shoemaker, Water Res. Research **43**, W01413 (2007).
13. M. Emmerich, A. H. Deutz and J. W. Klinkenberg, *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans 2011, IEEE (LA 2011), pp. 2147-2154.
14. D. Eriksson, D. Bindel and C. A. Shoemaker, Surrogate optimization toolbox, 2015, see https://github.com/dme65/pySOT.