Online Supplementary Material for

Reference Vector Assisted Candidate Search with Aggregated Surrogate Model for Expensive Many Objective Optimization Problems

Wenyu Wang[1] and Christine A. Shoemaker[1]

[1]*Department of Industrial Systems Engineering and Management, College of Design and Engineering, National University of Singapore, 1 Engineering Drive 2, Block E1A, Singapore.*

A Experimental Setup

In order to conduct fair comparison experiments, we set up the following global parameters for all algorithms. For test problems with 10 decision variables, the maximum number of expensive function evaluations is set to 300 and it is increased to 600 for test problems with 20 decision variables. Except for ε-MaSO, the other four algorithms are assisted by a set of uniformly distributed reference vectors (or reference points). To prevent generating excessive number of reference vectors, we use a two-layer strategy which implements the canonical simplex-lattice design method on boundary layer (with parameter $H = H_1$) and inside layer (with parameter $H = H_2$) simultaneously and combines their design results together (Cheng et al. 2016, Chugh et al. 2018, Deb and Jain 2014, Pan et al. 2019). Table 1 lists the parameter values of H_1 and H_2 and the corresponding number of reference vectors generated for different number of objective functions.

Table 1 Parameters used to generate reference vectors for problems with different number of objectives.

No. of objectives (k)	Parameters (H_1, H_2)	No. of reference vectors (n_r)
2	(49, 0)	50
3	(8, 0)	45
4	(4, 0)	35
6	(2, 2)	42
8	(2, 1)	44
10	(2, 0)	55

RECAS is implemented in Python and built upon the surrogate optimization toolbox pySOT (Eriksson et al. 2019). As illustrated in the algorithmic framework (see Section 3), RECAS adopts Latin Hypercube Design (McKay et al. 1979) to generate a total of $n_s = 11d - 1$ points at random for initialization. Within one iteration, the number of expensive objective function evaluations (n) is set to 5, which is the same as K-RVEA, MOEA/D-EGO, KTA2, and ParEGO. Consequently, RECAS aims to identify 5 center reference vectors and 5 center points accordingly. Around each center point, the number of candidates (i.e., q) is set to $100d$ with a minimum value of 500, which ensures a sufficient density in the vicinity of the center point (see Section 3.3.1). In addition, the upper bound and lower bound of the parameter σ associated with each point are respectively set to 0.2 and 0.2×0.5^5, i.e., $\sigma_{max} = 0.2$ and $\sigma_{min} = 0.2 \times 0.5^5$.

For alternative algorithms, we follow the parameter settings recommended by their original papers (Chugh et al. 2018, Deb and Jain 2014, Knowles 2006, Song et al. 2021, Wang et al. 2022, Zhang et al. 2010) and introduce some important parameters in the rest of this section. ε-MaSO is another surrogate-assisted many-objective optimization algorithm implemented in Python with pySOT toolbox. Unlike the other four algorithms, ε-MaSO is suggested to generate $2(d + 1)$ points via a symmetric Latin Hypercube Design for initialization, which saves more computational budget for optimization procedure. As a dominance-based method, ε-MaSO relies on the box-dominance, in which parameter ε plays a critical role. Therefore, as recommended in (Wang et al. 2022), the value of ε increases from 0.01 to 0.2 as the number of objectives increases.

For NSGA-III, K-RVEA, MOEA/D-EGO, KTA2, and ParEGO, we obtained their MATLAB implementations from the platform PlatEMO (Tian et al. 2017). All these three algorithms initially generate $11d - 1$ points at random via Latin Hypercube Design, which is the same as RECAS. Moreover, the size of their embedded population is the same as the number of reference vectors being constructed. In K-RVEA, the parameter used for determining whether diversity or convergence of reference vectors should be prioritized in selecting individuals is set to $0.05N$, where N denotes the population size, and the number of generations before updating the employed surrogate models is set to 20. The algorithm MOEA/D-EGO adopts $20 \times (11d - 1)$ fitness evaluations before updating its surrogates.

B Parameter Analysis

In this section, we conduct sensitive analysis on two important parameters used in RECAS, namely the number of initial points (i.e., n_s) and the batch size per iteration (i.e., n).

How does RECAS with different numbers of initial points perform? The decision on the value of n_s is primarily related to the balance between exploration and exploitation. Intuitively, initialization with a small number of points saves the budget for the iterative optimization procedure in RECAS; while initialization with a large number of points allows RECAS to explore the decision space in the early stage. We set the parameter n_s to $2d-1$, $5d-1$, $11d-1$, and $15d-1$, with d being the number of decision variables, and analyze their effects on the performance of RECAS on 10-dimensional DTLZ2 test problem within 300 function evaluations.

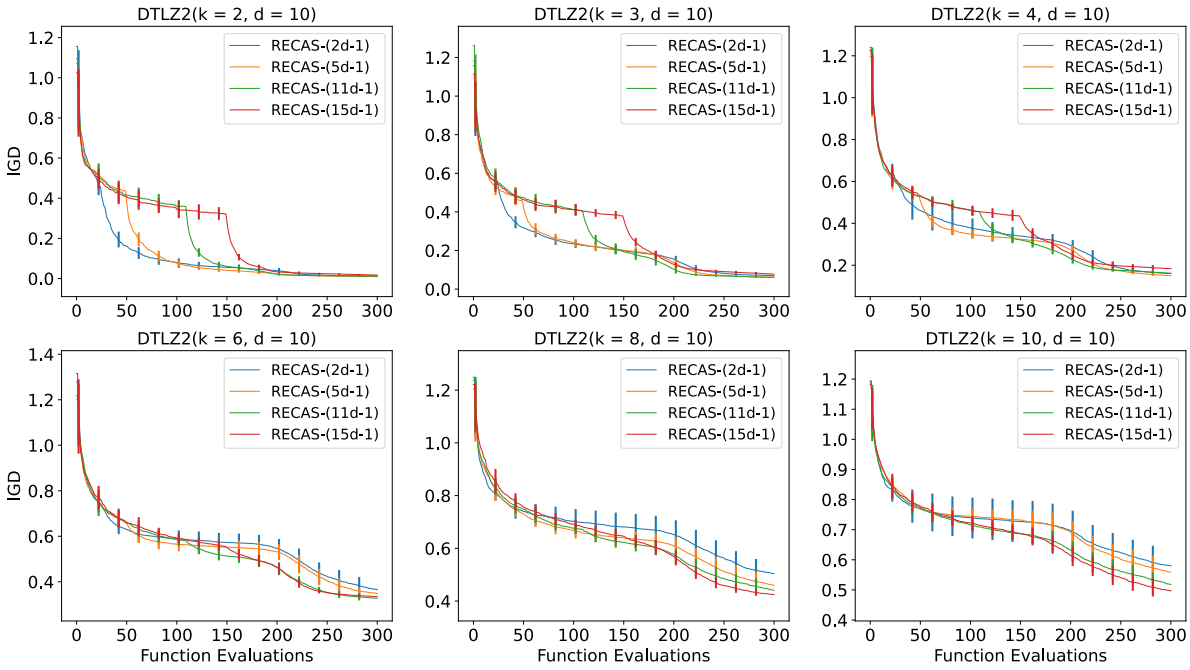

Figure 1 Average IGD progress curves (against the number of function evaluations) for RECAS with $2d-1$, $5d-1$, $11d-1$, *and* $15d-1$ *initial points on six DTLZ2 test problems.*

Figure 1 visualizes the average progress curves that plot the IGD indicator against the number of function evaluations for RECAS with different numbers of initial points. Generally, RECAS with less initial points generally starts the optimization procedure earlier and hence converges faster. However, as more objectives are considered, such an advantage becomes less obvious and RECAS significantly

benefits from the exhaust exploration of the decision space when starting with a larger number of initial points. Furthermore, in this study, RECAS adopts $11d - 1$ as the default value of n_s which was widely used in recent state-of-the-art multi-objective surrogate-assisted algorithms. It can be observed that compared with RECAS-($2d - 1$) and RECAS-($5d - 1$), RECAS-($11d - 1$) can always achieve better results after 300 function evaluations on DTLZ2 problems (except for the 4-objective case). RECAS with the option $15d - 1$ performs the worst on 2-, 3-, and 4-objective cases and such an option requires the algorithm to spend half of the computational budget into the initialization procedure and may not be an appropriate choice when a smaller budget is given.

How does RECAS with different batch sizes perform? In RECAS, since each surrogate-assisted candidate search determines one new points for expensive evaluation, the number of function evaluations conducted in each iteration, i.e., batch size (n), is equivalent to the number of center reference vectors or the number of center points. To experimentally elaborate the effects of n on the performance of RECAS, we run RECAS with n being 2, 5, 10, 15, and 20 on 10-dimensional DTLZ2 with a computational budget of 300 function evaluations. In Figure 2, we ignore the first $11d - 1$ initial sample points and plot IGD indicator against the number of iterations for RECAS with different values of n. Obviously, RECAS with a smaller n will run for more iterations.

In terms of the results obtained after 300 function evaluations, RECAS-5 performs the best on 4 out of 6 cases while RECAS-2 performs the best on the remaining two cases. The slightly worse performance of RECAS-10, RECAS-15, and RECAS-20 can be mainly attributed to the following two reasons: 1) They perform fewer iterations and therefore update surrogate models less frequently; 2) A large batch size means that they have to pick up a large number of center points, some of which may not be enough to guide an effective candidate search. However, in terms of iterations used to obtain the same IGD value, RECAS with a larger value of n shows greater efficiency. For example, on 4-objective DTLZ2, RECAS-2 and RECAS-5 take about 95 and 35 iterations, respectively, to reach the IGD value obtained by RECAS-20 after less than 10 iterations. Importantly, RECAS supports parallel processing as its candidate searches are independent to each other. Therefore, the batch size n can be set to the number of available processors to further improve the efficiency of RECAS.

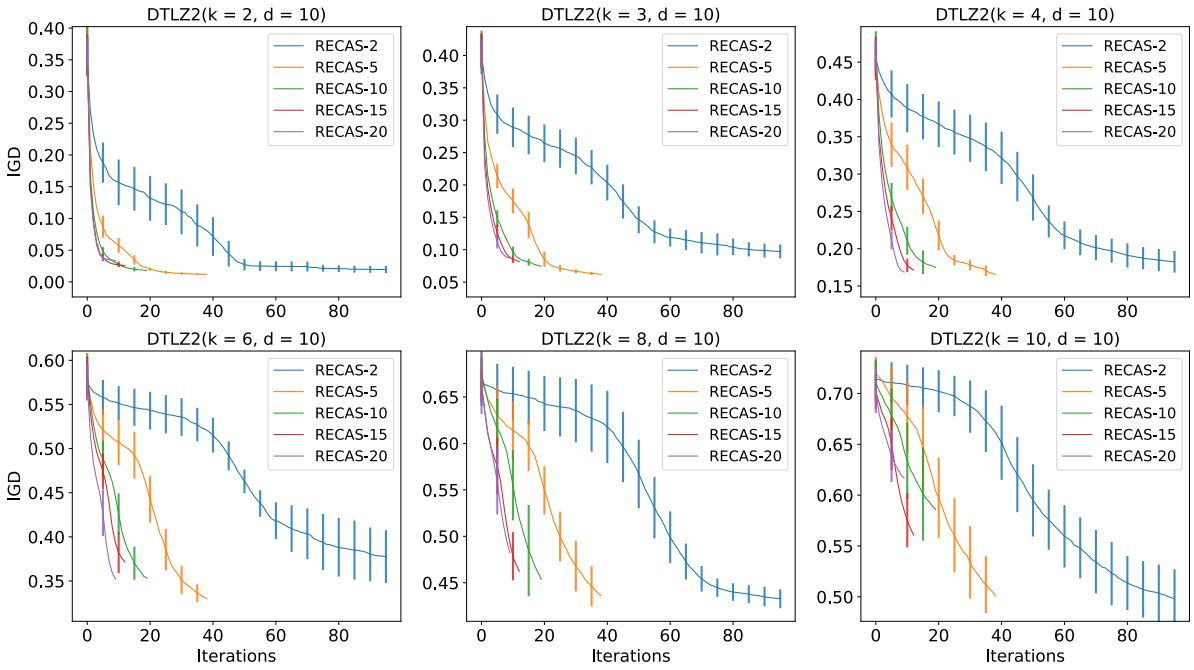

Figure 2 Average IGD progress curves (against the number of iterations) for RECAS with batch size being 2, 5, 10, 15, and 20 on six DTLZ2 test problems.

C Results on WFG Suite

The comparative experiments between RECAS and six other alternative algorithms are further conducted on the WFG test suite (from WFG1 to WFG9) (Huband et al. 2006). The performance of each selected algorithm is examined on WFG test problems with 2, 3, 4, 6, 8, and 10 objective functions. Table 2 records the average IGD values and Wilcoxon rank-sum test result (with respect to RECAS) of each algorithm and highlights the best numerical result of each case in grey. It can be observed that RECAS has the best IGD_{avg} on half of the 54 WFG cases and significant outperforms NSGA-III and MOEA/D-EGO on most cases.

WFG1 challenges whether the algorithms can maintain good diversity when dealing with biased density of points in the objective space and WFG2 has a disconnected Pareto-optimal front. The proposed algorithm RECAS does not perform well on these two test problems. Specifically, ParEGO achieves the best IGD_{avg} values and significantly dominates RECAS on all cases of WFG1. Meanwhile, on the different cases of WFG2, RECAS is outperformed by the different algorithms, among which K-RVEA and ε-MaSO show more stable and attractive performance.

Table 2 Numerical results of NSGA-III, K-RVEA, ε-MaSO, MOEA/D-EGO, KTA2, ParEGO, and RECAS over WFG1 to WFG7 where the number of objectives (k) varies from 2 to 10. For each algorithm, its IGD values obtained after evaluations are averaged on 20 independent trails and recorded in the "IGD_{avg}" column. The symbols "+", "−", and "≈" in the "T" column indicate that with 95% confidence level, the algorithm is significantly better, worse, or not statistically different to RECAS, respectively. In each row, the best (i.e., lowest) average IGD value is highlighted in grey. The last row summarizes the total number of symbols "+", "−", and "≈" for each algorithm.

Problem	No. of Obj.	NSGA-III		K-RVEA		ε-MaSO		MOEA/D-EGO		KTA2		ParEGO		RECAS
		IGD_{avg}	T	IGD_{avg}	T	IGD_{avg}	T	IGD_{avg}	T	IGD_{avg}	T	IGD_{avg}	T	IGD_{avg}
WFG1	2	1.99e+0	−	1.40e+0	+	1.61e+0	+	1.60e+0	≈	1.37e+0	+	1.30e+0	+	1.64e+0
	3	2.83e+0	−	1.87e+0	+	2.13e+0	+	2.16e+0	+	1.96e+0	+	1.56e+0	+	2.19e+0
	4	3.62e+0	−	2.61e+0	≈	2.72e+0	≈	2.82e+0	≈	2.60e+0	≈	1.92e+0	+	2.73e+0
	6	4.90e+0	−	3.67e+0	≈	3.82e+0	≈	3.98e+0	−	4.17e+0	≈	2.66e+0	+	3.71e+0
	8	6.45e+0	−	5.39e+0	≈	4.81e+0	≈	5.32e+0	≈	6.15e+0	≈	3.53e+0	+	4.89e+0
	10	8.63e+0	−	6.07e+0	≈	5.93e+0	≈	6.47e+0	−	8.37e+0	−	4.16e+0	+	6.25e+0
WFG2	2	5.16e-1	−	2.93e-1	+	4.58e-1	≈	3.97e-1	+	2.21e-1	+	4.47e-1	≈	4.43e-1
	3	6.73e-1	−	4.78e-1	+	5.34e-1	+	6.54e-1	−	8.17e-1	−	7.09e-1	−	6.39e-1
	4	8.69e-1	−	6.54e-1	+	7.27e-1	≈	8.59e-1	−	1.04e+0	−	8.82e-1	−	7.90e-1
	6	1.12e+0	≈	9.73e-1	+	9.52e-1	+	1.18e+0	−	1.39e+0	−	1.09e+0	≈	1.04e+0
	8	1.36e+0	−	1.33e+0	≈	1.26e+0	≈	1.38e+0	−	1.53e+0	−	1.10e+0	+	1.24e+0
	10	1.47e+0	−	1.70e+0	−	1.36e+0	≈	1.52e+0	−	1.63e+0	−	1.17e+0	+	1.37e+0
WFG3	2	4.97e-1	−	2.51e-1	−	8.25e-2	+	4.61e-1	−	1.92e-1	+	2.61e-1	−	2.10e-1
	3	6.07e-1	−	4.55e-1	−	2.48e-1	+	6.10e-1	−	3.43e-1	≈	5.09e-1	−	2.93e-1
	4	7.36e-1	−	5.87e-1	−	4.70e-1	−	7.77e-1	−	5.65e-1	−	5.58e-1	−	3.31e-1
	6	9.18e-1	−	8.01e-1	−	8.28e-1	−	1.00e+0	−	8.14e-1	−	5.77e-1	−	4.94e-1
	8	1.01e+0	−	8.74e-1	−	1.11e+0	−	1.13e+0	−	9.24e-1	−	6.18e-1	≈	5.67e-1
	10	8.56e-1	−	7.53e-1	−	1.03e+0	−	9.41e-1	−	9.02e-1	−	4.73e-1	≈	4.72e-1
WFG4	2	2.91e-1	−	2.78e-1	−	2.00e-1	−	2.76e-1	−	2.47e-1	−	2.83e-1	−	1.38e-1
	3	4.31e-1	−	4.37e-1	−	3.83e-1	−	4.55e-1	−	4.12e-1	−	4.58e-1	−	3.07e-1
	4	5.91e-1	−	6.16e-1	−	5.59e-1	−	6.23e-1	−	5.93e-1	−	6.21e-1	−	4.82e-1
	6	8.95e-1	−	9.36e-1	−	9.28e-1	−	9.73e-1	−	9.33e-1	−	9.37e-1	−	7.93e-1
	8	1.23e+0	−	1.29e+0	−	1.25e+0	−	1.33e+0	−	1.31e+0	−	1.29e+0	−	1.10e+0
	10	1.53e+0	−	1.62e+0	−	1.55e+0	−	1.62e+0	−	1.61e+0	−	1.53e+0	−	1.45e+0
WFG5	2	4.66e-1	−	1.81e-1	−	1.84e-1	−	2.21e-1	−	8.71e-2	+	8.78e-2	+	1.07e-1
	3	6.63e-1	−	3.22e-1	≈	4.43e-1	−	5.92e-1	−	3.23e-1	≈	4.27e-1	−	3.45e-1
	4	9.39e-1	−	7.02e-1	≈	7.59e-1	−	8.85e-1	−	6.61e-1	+	7.60e-1	−	6.98e-1
	6	1.51e+0	−	1.42e+0	−	1.33e+0	≈	1.43e+0	−	1.41e+0	−	1.31e+0	≈	1.30e+0
	8	2.09e+0	−	2.12e+0	−	1.91e+0	+	2.03e+0	−	2.05e+0	−	1.97e+0	≈	1.93e+0
	10	2.76e+0	−	2.89e+0	−	2.53e+0	+	2.60e+0	≈	2.75e+0	−	2.61e+0	≈	2.62e+0
WFG6	2	5.56e-1	−	4.99e-1	−	1.51e-1	+	4.08e-1	−	3.82e-1	−	5.86e-1	−	2.46e-1
	3	7.77e-1	−	7.02e-1	−	5.69e-1	−	7.77e-1	−	6.18e-1	−	7.56e-1	−	4.96e-1
	4	1.06e+0	−	9.91e-1	−	9.71e-1	−	1.08e+0	−	9.50e-1	−	9.90e-1	−	8.88e-1
	6	1.61e+0	−	1.59e+0	−	1.52e+0	−	1.65e+0	−	1.50e+0	−	1.56e+0	−	1.42e+0
	8	2.14e+0	−	2.25e+0	−	2.08e+0	−	2.25e+0	−	2.12e+0	−	2.24e+0	−	2.02e+0
	10	2.67e+0	−	2.90e+0	−	2.68e+0	−	2.78e+0	−	2.74e+0	−	2.57e+0	≈	2.59e+0
WFG7	2	4.21e-1	−	4.18e-1	−	2.45e-1	−	4.71e-1	−	3.64e-1	−	3.15e-1	−	1.47e-1
	3	6.42e-1	−	6.58e-1	−	5.47e-1	−	6.62e-1	−	5.68e-1	−	7.07e-1	−	3.87e-1
	4	9.61e-1	−	9.03e-1	≈	8.73e-1	≈	9.99e-1	−	8.95e-1	≈	1.05e+0	−	8.54e-1
	6	1.62e+0	−	1.56e+0	≈	1.52e+0	+	1.67e+0	−	1.52e+0	+	1.69e+0	−	1.58e+0
	8	2.29e+0	≈	2.30e+0	≈	2.16e+0	+	2.43e+0	−	2.19e+0	+	2.47e+0	−	2.31e+0
	10	3.10e+0	≈	2.99e+0	≈	2.86e+0	+	2.98e+0	≈	2.90e+0	+	3.15e+0	−	3.04e+0
WFG8	2	5.52e-1	−	4.24e-1	−	2.40e-1	+	5.14e-1	−	3.16e-1	≈	3.93e-1	−	3.02e-1
	3	7.99e-1	−	6.67e-1	−	5.34e-1	≈	8.06e-1	−	4.95e-1	+	7.60e-1	−	5.45e-1
	4	1.07e+0	−	9.59e-1	−	9.26e-1	≈	1.10e+0	−	8.61e-1	+	1.08e+0	−	8.98e-1
	6	1.57e+0	−	1.53e+0	−	1.50e+0	−	1.61e+0	−	1.44e+0	≈	1.62e+0	−	1.42e+0
	8	2.16e+0	−	2.18e+0	−	2.07e+0	≈	2.21e+0	−	2.12e+0	−	2.21e+0	−	2.05e+0
	10	2.75e+0	−	2.94e+0	−	2.75e+0	−	2.84e+0	−	2.85e+0	−	2.80e+0	−	2.65e+0
WFG9	2	4.84e-1	−	4.00e-1	−	3.39e-1	−	4.75e-1	−	3.47e-1	−	3.98e-1	−	2.96e-1
	3	6.77e-1	−	5.92e-1	−	5.85e-1	−	6.71e-1	−	5.83e-1	−	6.38e-1	−	4.27e-1
	4	9.26e-1	−	9.03e-1	−	9.38e-1	−	9.38e-1	−	7.04e-1	≈	8.76e-1	−	6.82e-1
	6	1.43e+0	−	1.41e+0	−	1.45e+0	−	1.45e+0	−	1.27e+0	≈	1.48e+0	−	1.20e+0
	8	1.97e+0	−	1.94e+0	≈	2.05e+0	−	2.06e+0	−	1.91e+0	≈	2.03e+0	−	1.82e+0
	10	2.53e+0	≈	2.61e+0	−	2.60e+0	−	2.57e+0	−	2.53e+0	≈	2.70e+0	−	2.48e+0
Number of +/−/≈		0/50/4		6/36/12		13/28/13		2/47/5		11/31/12		9/37/8		

On WFG3 which has a degenerated linear Pareto-optimal front, ε-MaSO performs the best on two- and three-objective cases while RECAS performs the best on the rest many-objective cases. WFG4 to WFG9 have the same concave Pareto-optimal front but they pose different challenges to algorithms,

including multimodal objectives (WFG4), deceptive objectives (WFG5), non-separable objectives (WFG6, WFG8 and WFG9). Obviously, RECAS dominates the other algorithms on all 12 cases of WFG4 and WFG9. It is worthy of noting that RECAS shows the best performance on low-dimensional objective space for WFG7 and high-dimensional objective space for WFG6 and WFG8. Finally, for WFG5, K-RVEA, ε-MaSO, KTA2, ParEGO, and RECAS are generally competitive to each other.

D Results on Watershed Model Calibration Problems

We examine the potential of RECAS to handle real-world applications through calibrating the Townbrook watershed simulation model (Tolson and Shoemaker 2007). Townbrook is a sub-watershed of the Cannonsville watershed in upstate New York, USA. Flows from the entire watershed are eventually stored in the Cannonsville reservoir that supplies drinking water to New York City. Phosphorous loads from Cannonsville watershed into the reservoir need to be carefully monitored because excess phosphorous can cause algal growth that degrades water quality and makes water treatment more expensive. Based on the spatially distributed watershed model SWAT (Arnold et al. 1998), Tolson and Shoemaker (Tolson and Shoemaker 2007) developed a complex simulation model for Townbrook watershed to predict the flows, sediment, nutrients, and other contaminant transported into the reservoir. In this paper, we are interested in calibrating a set of 15 parameters of Townbrook watershed model for the sake of making more accurate prediction and assessment on the water quality.

As pointed out in the prior studies, multiple objectives should be simultaneously considered during the calibration of watershed simulation models (Ahmadi et al. 2014, Akhtar and Shoemaker 2016). Therefore, two multi-objective model calibration problems are formulated, namely TBrook1 (in which $k = 3, d = 15$) and TBrook2 (in which $k = 4, d = 15$), where different measures of error between simulation result and observed data are adopted as the calibration objectives, including correlation, bias, relative variability and Box-Cox Transformed NSE (Willems 2009). Since Townbrook is a relatively small watershed, a single simulation run of Townbrook watershed model over a ten-year historical flow time-series data set takes time in the order of seconds. However, it still takes hours to conduct tens of thousands of function evaluations on both TBrook1 and TBrook2. In sum, TBrook1 is a computationally expensive MOP while TBrook2 is a computationally expensive MaOP.

RECAS together with K-RVEA, ε-MaSO and KTA2 that have good performances on the synthetic test problems (see Section C and Section 5.2 in the main paper) are applied to TBrook1 and TBrook2. The computational budget is set to 600 function evaluations and each algorithm runs 20 independent trials for each problem. Note that since the true Pareto-optimal fronts of TBrook1 and TBrook2 are unknown, when calculating the IGD indicator, we construct an ideal front to approximate the Pareto-optimal front by identifying and collecting the non-dominated objective vectors from the independent runs of all algorithms. Considering that the fact of using the ideal fronts for IGD calculation may mis-lead the analysis on real-world applications, besides IGD, we additionally report Hypervolume (HV, which measures the volume occupied by a non-dominated front (Zitzler and Thiele 1999)) and number of non-dominated points (NS) located by each algorithm after 600 function evaluations through box plots in Figure 3. All of three indicators have been rescaled to $[0, 1]$ where the worst and best values are respectively denoted by 0 and 1 and hence, a higher distribution of box plot is preferred.

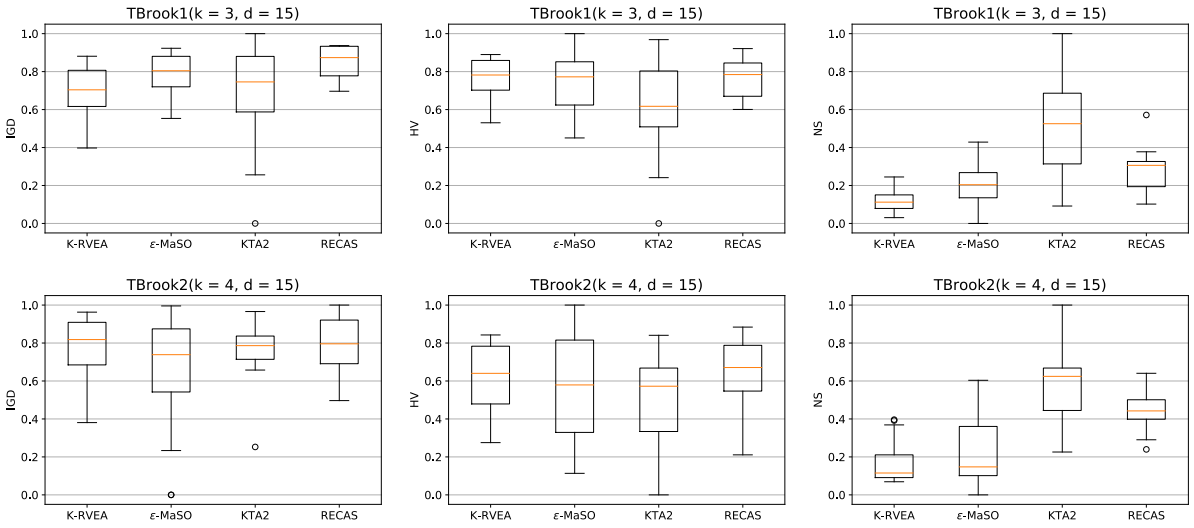

Figure 3 Box plots of IGD, HV and number of non-dominated solutions (NS) obtained by K-RVEA, ε-MaSO, KTA2, and RE-CAS after 600 function evaluations on TBrook1 (in the first row) and TBrook2 (in the second row). ***All of three indicators have been rescaled to the interval*** $[\mathbf{0}, \mathbf{1}]$***, where*** $\mathbf{0}$ ***and*** $\mathbf{1}$ ***represent the worst and best indicator values, respectively.***

It can be observed from Figure 3 that on both TBrook1 and TBrook2, the median IGD and HV indicators of RECAS are generally the best (i.e., highest) among all algorithms, which shows the effective performance of RECAS. Although both IGD and HV indicators measure the algorithmic performance regarding to convergence and diversity, their ranking outcomes for the four surrogate-assisted optimization algorithms are different and it is hard to tell which algorithm is significantly better

than the others. For example, although the best HV values of ε-MaSO are the highest, its worst HV values are greatly lower than K-RVEA and RECAS. In other words, the performance of MaSO in terms of HV indicator is less stable than the others. Finally, KTA2 outperforms the others in terms of NS indicator and it indicates that KTA2 is able to generate more trade-off points on the non-dominated front found by it. Since the embedded simulation model of TBrook1 and TBrook2 is complicated in nature and the performance of RECAS is comparable to the other three state-of-the-art algorithms, we believe that RECAS is able to handle computationally expensive many-objective real-world applications in an effective manner.

References

Ahmadi M, Arabi M, Ascough JC, Fontane DG, Engel BA (2014) Toward improved calibration of watershed models: Multisite multiobjective measures of information. *Environmental Modelling and Software* 59:135–145.

Akhtar T, Shoemaker CA (2016) Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection. *Journal of Global Optimization* 64:17–32.

Arnold JG, Srinivasan R, Muttiah RS, Williams JR (1998) Large area hydrologic modeling and assessment part I: Model development. *Journal of the American Water Resources Association* 34(1):73–89.

Cheng R, Jin Y, Olhofer M, Sendhoff B, Member S (2016) A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 20(5):773–791.

Chugh T, Jin Y, Miettinen K, Hakanen J, Sindhya K (2018) A Surrogate-Assisted Reference Vector Guided Evolutionary Algorithm for Computationally Expensive Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 22(1):129–142.

Deb K, Jain H (2014) An Evolutionary Many-objective Optimization Algorithm Using Reference-point-based Nondominated Sorting Approach, Part I: Solving Problems with Box Constraints. *IEEE Transactions on Evolutionary Computation* 18(4):577–601.

Eriksson D, Bindel D, Shoemaker CA (2019) pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. *arXiv:1908.00420*.

Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation* 10(5):477–506.

Knowles J (2006) ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 10(1):50–66.

McKay MD, Beckman RJ, Conover WJ (1979) A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* 21(2):239–245.

Pan L, He C, Tian Y, Wang H, Zhang X, Jin Y (2019) A Classification-Based Surrogate-Assisted Evolutionary Algorithm for Expensive Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 23(1):74–88.

Song Z, Wang H, He C, Jin Y (2021) A Kriging-Assisted Two-Archive Evolutionary Algorithm for Expensive Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 25(6):1013–1027.

Tian Y, Cheng R, Zhang X, Jin Y (2017) PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization. *IEEE Computational Intelligence Magazine* 12(4):73–87.

Tolson BA, Shoemaker CA (2007) Cannonsville Reservoir Watershed SWAT2000 model development, calibration and validation. *Journal of Hydrology* 337(1–2):68–86.

Wang W, Akhtar T, Shoemaker CA (2022) Integrating ε-dominance and RBF surrogate optimization for solving computationally expensive many-objective optimization problems. *Journal of Global Optimization* 82:965–992.

Willems P (2009) A time series tool to support the multi-criteria performance evaluation of rainfall-runoff models. *Environmental Modelling and Software* 24(3):311–321.

Zhang Q, Liu W, Tsang E, Virginas B (2010) Expensive multiobjective optimization by MOEA/D with gaussian process model. *IEEE Transactions on Evolutionary Computation* 14(3):456–474.

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3(4):257–271.